

# INTERGRAPH

---

May 12, 1985

STANDARD INTERCHANGE  
FORMAT (SIF) COMMAND  
LANGUAGE  
IMPLEMENTATION GUIDE  
(8.8)

DIXD4110

SLGR611

Intergraph Corporation  
One Madison Industrial Park  
Huntsville, Alabama 35807  
Phone: (205) 772-2000  
TWX 810-726-2180

---

The information and the software described in this document are subject to change without notice and should not be construed as commitments by Intergraph Corporation. Intergraph Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Intergraph or its affiliated companies.

Intergraph is a registered trademark of Intergraph Corporation. DMRS, File Processor, IQDS, Interact, Intermap, Interpage, Interpro, Interpro 32, Interview, and Micro II are trademarks of Intergraph Corporation.

DEC and VAX are registered trademarks of Digital Equipment Corporation.

## PREFACE

This document supports the 8.8 software release version at the Standard Interchange Format (SIF) and is intended to describe the method by which SIF is implemented. It replaces the Intergraph Standard Interchange Format (ISIF) Command Language Implementation (DIXD1300). SIF supports both 2-D and 3-D graphics and is available on the VAX system. SIF files are transferred between systems via magnetic tape. The SIF product provides the software to read and write these tapes in a SIF format. SIF is useful to anyone who has an application for transferring graphics files and an associated database between systems.





## TABLE OF CONTENTS

<u>Section</u>	<u>Title</u>	<u>Page</u>
1.	INTRODUCTION.....	1-1
1.1	SIF Processors.....	1-1
2.	SIF ENVIRONMENT FILE.....	2-1
2.1	AR - Define the ARC Command Format for SIF-out....	2-1
2.2	AS - Define the SIF ASCII File Name.....	2-3
2.3	BY - Define the Byte Storage for SIF Binary Tapes.....	2-4
2.4	CL - Define the IGDS Cell Library File.....	2-5
2.5	CD - Define the Integer Representation for SIF Binary Tapes.....	2-9
2.6	CZ - Define the Constant Z for 2-D to 3-D SIF Translations.....	2-9
2.7	DB - Define the DMRS Database File Name.....	2-9
2.8	DE - Define the Debug Processing Option.....	2-10
2.9	DF - Define the IGDS Design File Name.....	2-10
2.10	DM - Define the Print Attribute Data Option.....	2-11
2.11	DP - Define the Drop Paragraph Option.....	2-11
2.12	DU - Define the Design File Working Units.....	2-12
2.13	ED - Define the Process Enter Data Field Option...	2-13
2.14	EL - Define the Error Level.....	2-14
2.15	EM - Define the SIF Message File.....	2-14
2.16	ER - Define the Recovery Option.....	2-15
2.17	ET - Define the EBCDIC Tape Format Option.....	2-16
2.18	FE - Define the Fence for SIF-out Processing.....	2-16



## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
2.19	FL - Define the IQDS Font Library File.....	2-17
2.20	FS - Define the IQDS Design File Size.....	2-17
2.21	FT - Define the Font Processing Option.....	2-18
2.22	IN - Define the 32-bit Integer Storage for SIF Binary Tapes.....	2-18
2.23	IX - Define the IQDS Map Index File Name.....	2-20
2.24	LR - Define the SIF ASCII Logical Record Length...	2-21
2.25	MA - Define the Generate Matrix Option.....	2-22
2.26	MD - Define the Graphic Mode.....	2-23
2.27	MT - Define the Tape Drive.....	2-23
2.28	NE - Define the Negative Coordinate Option.....	2-23
2.29	OV - Define the Overlay Processing Option.....	2-24
2.30	PF - Define the IQDS Pattern File Name.....	2-25
2.31	PT - Define the Pad Text Option.....	2-28
2.32	RA - Define the RANGE Command Option.....	2-29
2.33	SD - Define the IQDS Seed File Name.....	2-29
2.34	SX - Define the SIF Binary File Name.....	2-30
2.35	TC - Define the Track Changes Option.....	2-31
2.36	TD - Define the Tape Density.....	2-32
2.37	TF - Define the Tape Files to Process.....	2-32
2.38	TN - Define the PARAGRAPH Command Format for SIF-out.....	2-33
2.39	TX - Define the TEXT Command Format for SIF-out.....	2-34
2.40	TY - Define the Element Type Processing Option....	2-34



## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
3.	SIF INDEX FILE.....	3-1
4.	SIF ASCII COMMAND LANGUAGE.....	4-1
4.1	Special SIF ASCII Commands.....	4-1
4.1.1	MATRIX Command (MTX/) .....	4-2
4.1.2	GENERATE PARAGRAPH RANGE BLOCK Command (PA3/).....	4-3
4.1.3	GENERATE TEXT LINE RANGE BLOCK Command (TX3/).....	4-3
5.	SIF BINARY COMMAND LANGUAGE.....	5-1
5.1	SIF Coordinate Format.....	5-2
5.2	SIF Floating Point Format.....	5-3
6.	SIF INTERFACE LIBRARIES.....	6-1
6.1	Image Linking.....	6-1
6.2	File Maintenance.....	6-2
6.2.1	SIFOPEN - Open SIF File.....	6-2
6.2.2	SIFEOF - End SIF File.....	6-3
6.2.3	SIFCLO - Close SIF File.....	6-3
6.3	File Characteristics.....	6-3
6.3.1	SIFCOL - ASCII Characters Per Record.....	6-3
6.3.2	SIFMOD - Define Graphic Mode.....	6-3
6.3.3	SIFNEG - Define Coordinate Format.....	6-4
6.4	Utility Subroutines.....	6-4
6.4.1	SIFATT - Define Attribute Value.....	6-4
6.4.2	SIFMAT - Compute Transformation Matrix.....	6-5
6.4.3	SIFPTS - Define Vertices.....	6-5
6.4.4	SIFTND - Define Paragraph Boundary.....	6-6
6.4.5	SIFTXT - Define Text Line Boundary.....	6-7
6.5	Internal Subroutines.....	6-7
7.	GRAPHIC CHARACTERISTICS COMMANDS.....	7-1
7.1	OVERLAY Command.....	7-1



# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
7.1.1	OVERLAY Command - ASCII Form.....	7-2
7.1.2	OVERLAY Command - Binary Form.....	7-2
7.1.3	OVERLAY Command - Interface Form.....	7-2
7.1.4	OVERLAY Command Restrictions.....	7-3
7.1.5	OVERLAY Command Examples.....	7-3
7.2	ACTIVE Z Command.....	7-4
7.2.1	ACTIVE Z - ASCII Form.....	7-5
7.2.2	ACTIVE Z Command - Binary Form.....	7-5
7.2.3	ACTIVE Z Command - Interface Form.....	7-6
7.2.4	ACTIVE Z Command Restrictions.....	7-6
7.2.5	ACTIVE Z Command Examples.....	7-6
7.3	CLASSIFICATION Command.....	7-7
7.3.1	CLASSIFICATION Command - ASCII Form.....	7-8
7.3.2	CLASSIFICATION Command - Binary Form.....	7-8
7.3.3	CLASSIFICATION Command - Interface Form.....	7-8
7.3.4	CLASSIFICATION Command Restrictions.....	7-9
7.3.5	CLASSIFICATION Command Examples.....	7-9
7.4	ASSOCIATION Command.....	7-11
7.4.1	ASSOCIATION Command - ASCII Form.....	7-11
7.4.2	ASSOCIATION Command - Binary Form.....	7-12
7.4.3	ASSOCIATION Command - Interface Form.....	7-12
7.4.4	ASSOCIATION Command Restrictions.....	7-12
7.4.5	ASSOCIATION Command Examples.....	7-13
7.5	FONT Command.....	7-14
7.5.1	FONT Command - ASCII Form.....	7-14
7.5.2	FONT Command - Binary Form.....	7-14
7.5.3	FONT Command - Interface Form.....	7-15
7.5.4	FONT Command Restrictions.....	7-15
7.5.5	FONT Command Examples.....	7-15
7.6	PATTERN Command.....	7-17
7.6.1	PATTERN Command - ASCII Form.....	7-18
7.6.2	PATTERN Command - Binary Form.....	7-18
7.6.3	PATTERN Command - Interface Form.....	7-19
7.6.4	PATTERN Command Restrictions.....	7-20
7.6.5	PATTERN Command Examples.....	7-20
7.7	LINE CHARACTERISTICS Command.....	7-23
7.7.1	LINE CHARACTERISTICS Command - ASCII Form.....	7-24
7.7.2	LINE CHARACTERISTICS Command - Binary Form.....	7-25
7.7.3	LINE CHARACTERISTICS Command - Interface Form.....	7-25
7.7.4	LINE CHARACTERISTICS Command Restrictions.....	7-26
7.7.5	LINE CHARACTERISTICS Command Examples.....	7-26





## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
7.8	TEXT CHARACTERISTICS Command.....	7-30
7.8.1	TEXT CHARACTERISTICS Command - ASCII Form.....	7-31
7.8.2	TEXT CHARACTERISTICS Command - Binary Form.....	7-31
7.8.3	TEXT CHARACTERISTICS Command - Interface Form.....	7-32
7.8.4	TEXT CHARACTERISTICS Command Restrictions.....	7-32
7.8.5	TEXT CHARACTERISTICS Command Examples.....	7-32
7.9	PARAGRAPH CHARACTERISTICS Command.....	7-34
7.9.1	PARAGRAPH CHARACTERISTICS Command - ASCII Form....	7-34
7.9.2	PARAGRAPH CHARACTERISTICS Command - Binary Form... 7-35	
7.9.3	PARAGRAPH CHARACTERISTICS Command - Interface Form.....	7-36
7.9.4	PARAGRAPH CHARACTERISTICS Command Restrictions....	7-37
7.9.5	PARAGRAPH CHARACTERISTICS Command Examples.....	7-37
7.10	TEMPORARY ORIGIN Command.....	7-39
7.10.1	TEMPORARY ORIGIN Command - ASCII Form.....	7-39
7.10.2	TEMPORARY ORIGIN Command - Binary Form.....	7-40
7.10.3	TEMPORARY ORIGIN Command - Interface Form.....	7-40
7.10.4	TEMPORARY ORIGIN Command Restrictions.....	7-41
7.10.5	TEMPORARY ORIGIN Command Examples.....	7-41
7.11	IDENTIFIER Command.....	7-43
7.11.1	IDENTIFIER Command - ASCII Form.....	7-45
7.11.2	IDENTIFIER Command - Binary Form.....	7-45
7.11.3	IDENTIFIER Command - Interface Form.....	7-46
7.11.4	IDENTIFIER Command Restrictions.....	7-46
7.11.5	IDENTIFIER Command Examples.....	7-46
7.12	MULTIPLE IDENTIFIER Command.....	7-47
7.12.1	MULTIPLE IDENTIFIER Command - ASCII Form.....	7-48
7.12.2	MULTIPLE IDENTIFIER Command - Binary Form.....	7-48
7.12.3	MULTIPLE IDENTIFIER Command - Interface Form.....	7-49
7.12.4	MULTIPLE IDENTIFIER Command Restrictions.....	7-49
7.12.5	MULTIPLE IDENTIFIER Command Examples.....	7-50
8.	GRAPHIC ELEMENT GENERATION COMMANDS.....	8-1
8.1	LINE STRING Command - Generate Line, Line String, Shape.....	8-1
8.1.1	LINE STRING Command - ASCII Form.....	8-2
8.1.2	LINE STRING Command - Binary Form.....	8-2
8.1.3	LINE STRING Command - Interface Form.....	8-3
8.1.4	LINE STRING Command Restrictions.....	8-4
8.1.5	LINE STRING Command Examples.....	8-4



# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
8.2	CIRCLE Command - Generate Circle.....	8-7
8.2.1	CIRCLE Command - ASCII Form.....	8-8
8.2.2	CIRCLE Command - Binary Form.....	8-8
8.2.3	CIRCLE Command - Interface Form.....	8-10
8.2.4	CIRCLE Command Restrictions.....	8-11
8.2.5	CIRCLE Command Examples.....	8-11
8.3	ARC Command - Generate Arc.....	8-13
8.3.1	ARC Command - ASCII Form.....	8-14
8.3.2	ARC Command - Binary Form.....	8-15
8.3.3	ARC Command - Interface Form.....	8-17
8.3.4	ARC Command Restrictions.....	8-17
8.3.5	ARC Command Examples.....	8-18
8.4	SYMBOL Command - Generate Cell.....	8-21
8.4.1	SYMBOL Command - ASCII Form.....	8-22
8.4.2	SYMBOL Command - Binary Form.....	8-22
8.4.3	SYMBOL Command - Interface Form.....	8-23
8.4.4	SYMBOL Command Restrictions.....	8-23
8.4.5	SYMBOL Command Examples.....	8-24
8.5	INCLUDE TEXT Command - Enter Data Fill-In.....	8-26
8.5.1	INCLUDE TEXT Command - ASCII Form.....	8-27
8.5.2	INCLUDE TEXT Command - Binary Form.....	8-27
8.5.3	INCLUDE TEXT Command - Interface Form.....	8-28
8.5.4	INCLUDE TEXT Command Restrictions.....	8-28
8.5.5	INCLUDE TEXT Command Examples.....	8-29
8.6	CURVE Command - Generate Curve.....	8-30
8.6.1	CURVE Command - ASCII Form.....	8-30
8.6.2	CURVE Command - Binary Form.....	8-30
8.6.3	CURVE Command - Interface Form.....	8-31
8.6.4	CURVE Command Restrictions.....	8-31
8.6.5	CURVE Command Examples.....	8-32
8.7	ELLIPSE Command - Generate Ellipse.....	8-34
8.7.1	ELLIPSE Command - ASCII Form.....	8-34
8.7.2	ELLIPSE Command - Binary Form.....	8-35
8.7.3	ELLIPSE Command - Interface Form.....	8-35
8.7.4	ELLIPSE Command Restrictions.....	8-36
8.7.5	ELLIPSE Command Examples.....	8-37



# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
8.8	ELLIPTICAL ARC Command - Generate Partial Ellipse.....	8-40
8.8.1	ELLIPTICAL ARC Command - ASCII Form.....	8-40
8.8.2	ELLIPTICAL ARC Command - Binary Form.....	8-41
8.8.3	ELLIPTICAL ARC Command - Interface Form.....	8-41
8.8.4	ELLIPTICAL ARC Command Restrictions.....	8-43
8.8.5	ELLIPTICAL ARC Command Examples.....	8-43
8.9	CONIC Command - Generate Conic.....	8-47
8.9.1	CONIC Command - ASCII Form.....	8-47
8.9.2	CONIC Command - Binary Form.....	8-47
8.9.3	CONIC Command - Interface Form.....	8-48
8.9.4	CONIC Command Restrictions.....	8-48
8.9.5	CONIC Command Examples.....	8-49
8.10	BEGIN COMPLEX STRING Command - Generate Complex String or Complex Shape.....	8-49
8.10.1	BEGIN COMPLEX STRING Command - ASCII Form.....	8-51
8.10.2	BEGIN COMPLEX STRING Command - Binary Form.....	8-51
8.10.3	BEGIN COMPLEX STRING Command - Interface Form.....	8-51
8.10.4	BEGIN COMPLEX STRING Command Restrictions.....	8-52
8.10.5	BEGIN COMPLEX STRING Command Examples.....	8-52
8.11	END COMPLEX STRING Command - Generate Complex String or Complex Shape.....	8-53
8.11.1	END COMPLEX STRING Command - ASCII Form.....	8-54
8.11.2	END COMPLEX STRING Command - Binary Form.....	8-54
8.11.3	END COMPLEX STRING Command - Interface Form.....	8-54
8.11.4	END COMPLEX STRING Command Restrictions.....	8-54
8.11.5	END COMPLEX STRING Command Examples.....	8-54
8.12	BEGIN SYMBOL Command - Expanded Cell; Orphan Cell.....	8-54
8.12.1	BEGIN SYMBOL Command - ASCII Form.....	8-55
8.12.2	BEGIN SYMBOL Command - Binary Form.....	8-55
8.12.3	BEGIN SYMBOL Command - Interface Form.....	8-56
8.12.4	BEGIN SYMBOL Command Restrictions.....	8-56
8.12.5	BEGIN SYMBOL Command Examples.....	8-57
8.13	END SYMBOL Command.....	8-59
8.13.1	END SYMBOL Command - ASCII Form.....	8-59
8.13.2	END SYMBOL Command - Binary Form.....	8-59
8.13.3	END SYMBOL Command - Interface Form.....	8-59
8.13.4	END SYMBOL Command Restrictions.....	8-59
8.13.5	END SYMBOL Command Examples.....	8-59



# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
8.14	BEGIN SURFACE Command - Generate Surface Element.....	8-60
8.14.1	BEGIN SURFACE Command - ASCII Form.....	8-60
8.14.2	BEGIN SURFACE Command - Binary Form.....	8-60
8.14.3	BEGIN SURFACE Command - Interface Form.....	8-60
8.14.4	BEGIN SURFACE Command Restrictions.....	8-61
8.14.5	BEGIN SURFACE Command Examples.....	8-61
8.15	END SURFACE Command.....	8-65
8.15.1	END SURFACE Command - ASCII Form.....	8-65
8.15.2	END SURFACE Command - Binary Form.....	8-65
8.15.3	END SURFACE Command - Interface Form.....	8-65
8.15.4	END SURFACE Command Restrictions.....	8-65
8.15.5	END SURFACE Command Examples.....	8-65
8.16	BEGIN SOLID Command - Generate Capped Surface Element.....	8-66
8.16.1	BEGIN SOLID Command - ASCII Form.....	8-66
8.16.2	BEGIN SOLID Command - Binary Form.....	8-66
8.16.3	BEGIN SOLID Command - Interface Form.....	8-66
8.16.4	BEGIN SOLID Command Restrictions.....	8-67
8.16.5	BEGIN SOLID Command Examples.....	8-67
8.17	END SOLID Command.....	8-69
8.17.1	END SOLID Command - ASCII Form.....	8-69
8.17.2	END SOLID Command - Binary Form.....	8-69
8.17.3	END SOLID Command - Interface Form.....	8-69
8.17.4	END SOLID Command Restrictions.....	8-69
8.17.5	END SOLID Command Examples.....	8-69
8.18	POINT STRING Command - Generate Point String.....	8-70
8.18.1	POINT STRING Command - ASCII Form.....	8-70
8.18.2	POINT STRING Command - Binary Form.....	8-71
8.18.3	POINT STRING Command - Interface Form.....	8-72
8.18.4	POINT STRING Command Restrictions.....	8-72
8.18.5	POINT STRING Command Examples.....	8-73
8.19	CYLINDER Command - Generate Circular Cylinder.....	8-76
8.19.1	CYLINDER Command - ASCII Form.....	8-76
8.19.2	CYLINDER Command - Binary Form.....	8-76
8.19.3	CYLINDER Command - Interface Form.....	8-79
8.19.4	CYLINDER Command Restrictions.....	8-79
8.19.5	CYLINDER Command Examples.....	8-80





# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
8.20	CIRCULAR TRUNCATED CONE Command - Generate Circular Truncated Cone.....	8-82
8.20.1	CIRCULAR TRUNCATED CONE Command - ASCII Form.....	8-82
8.20.2	CIRCULAR TRUNCATED CONE Command - Binary Form.....	8-84
8.20.3	CIRCULAR TRUNCATED CONE Command - Interface Form.....	8-86
8.20.4	CIRCULAR TRUNCATED CONE Command Restrictions.....	8-86
8.20.5	CIRCULAR TRUNCATED CONE Command Examples.....	8-87
8.21	BEGIN B-SPLINE CURVE Command.....	8-89
8.21.1	BEGIN B-SPLINE CURVE Command - ASCII Form.....	8-91
8.21.2	BEGIN B-SPLINE CURVE Command - Binary Form.....	8-91
8.21.3	BEGIN B-SPLINE CURVE Command - Interface Form.....	8-92
8.21.4	BEGIN B-SPLINE CURVE Command Restrictions.....	8-92
8.21.5	BEGIN B-SPLINE CURVE Command Examples.....	8-93
8.22	END B-SPLINE CURVE Command.....	8-100
8.22.1	END B-SPLINE CURVE Command - ASCII Form.....	8-100
8.22.2	END B-SPLINE CURVE Command - Binary Form.....	8-100
8.22.3	END B-SPLINE CURVE Command - Interface Form.....	8-100
8.22.4	END B-SPLINE CURVE Command Restrictions.....	8-100
8.22.5	END B-SPLINE CURVE Command Examples.....	8-100
8.23	BEGIN B-SPLINE SURFACE Command.....	8-100
8.23.1	BEGIN B-SPLINE SURFACE Command - ASCII Form.....	8-102
8.23.2	BEGIN B-SPLINE SURFACE Command - Binary Form.....	8-103
8.23.3	BEGIN B-SPLINE SURFACE Command - Interface Form...	8-104
8.23.4	BEGIN B-SPLINE SURFACE Command Restrictions.....	8-104
8.23.5	BEGIN B-SPLINE SURFACE Command Examples.....	8-105
8.24	END B-SPLINE SURFACE Command.....	8-110
8.24.1	END B-SPLINE SURFACE Command - ASCII Form.....	8-110
8.24.2	END B-SPLINE SURFACE Command - Binary Form.....	8-110
8.24.3	END B-SPLINE SURFACE Command - Interface Form.....	8-110
8.24.4	END B-SPLINE SURFACE Command Restrictions.....	8-110
8.24.5	END B-SPLINE SURFACE Command Examples.....	8-110
8.25	B-spline Component Elements.....	8-110
8.26	BOUNDARY Command.....	8-111
8.26.1	BOUNDARY Command - ASCII Form.....	8-111
8.26.2	BOUNDARY Command - Binary Form.....	8-111
8.26.3	BOUNDARY Command - Interface Form.....	8-112



# TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
8.27	KNOT Command.....	8-112
8.27.1	KNOT Command - ASCII Form.....	8-112
8.27.2	KNOT Command - Binary Form.....	8-113
8.27.3	KNOT Command - Interface Form.....	8-113
8.28	POLE Command.....	8-114
8.28.1	POLE Command - ASCII Form.....	8-114
8.28.2	POLE Command - Binary Form.....	8-114
8.28.3	POLE Command - Interface Form.....	8-115
8.29	WEIGHT Command.....	8-115
8.29.1	WEIGHT Command - ASCII Form.....	8-115
8.29.2	WEIGHT Command - Binary Form.....	8-115
8.29.3	WEIGHT Command - Interface Form.....	8-116
9.	GRAPHIC TEXT GENERATION COMMANDS.....	9-1
9.1	TEXT LINE Command - Generate Text String.....	9-1
9.1.1	TEXT LINE Command - ASCII Form.....	9-2
9.1.2	TEXT LINE Command - Binary Form.....	9-3
9.1.3	TEXT LINE Command - Interface Form.....	9-4
9.1.4	TEXT LINE Command Restrictions.....	9-5
9.1.5	TEXT LINE Command Examples.....	9-5
9.2	PARAGRAPH Command - Generate Text Node.....	9-10
9.2.1	PARAGRAPH Command - ASCII Form.....	9-11
9.2.2	PARAGRAPH Command - Binary Form.....	9-13
9.2.3	PARAGRAPH Command - Interface Form.....	9-14
9.2.4	PARAGRAPH Command Restrictions.....	9-15
9.2.5	PARAGRAPH Command Examples.....	9-15
9.3	GENERATE PARAGRAPH LINE Command - Text Node Line .....	9-16
9.3.1	GENERATE PARAGRAPH LINE Command - ASCII Form.....	9-18
9.3.2	GENERATE PARAGRAPH LINE Command - Binary Form.....	9-18
9.3.3	GENERATE PARAGRAPH LINE Command - Interface Form.....	9-18
9.3.4	GENERATE PARAGRAPH LINE Command Restrictions.....	9-19
9.3.5	GENERATE PARAGRAPH LINE Command Examples.....	9-19



## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
9.4	CLOSE PARAGRAPH - End Text Node.....	9-21
9.4.1	CLOSE PARAGRAPH - ASCII Form.....	9-21
9.4.2	CLOSE PARAGRAPH - Binary Form.....	9-21
9.4.3	CLOSE PARAGRAPH - Interface Form.....	9-21
9.4.4	CLOSE PARAGRAPH Command Restrictions.....	9-21
9.4.5	CLOSE PARAGRAPH Command Examples.....	9-21
10.	MISCELLANEOUS COMMANDS.....	10-1
10.1	PAD - No Operation.....	10-1
10.1.1	PAD Command - ASCII Form.....	10-1
10.1.2	PAD Command - Binary Form.....	10-1
10.1.3	PAD Command - Interface Form.....	10-1
10.1.4	PAD Command Restrictions.....	10-2
10.1.5	PAD Command Examples.....	10-2
10.2	GRAPHIC ASSOCIATION DESCRIPTOR Command - Group Data.....	10-2
10.2.1	GRAPHIC ASSOCIATION DESCRIPTOR Command - ASCII Form.....	10-3
10.2.2	GRAPHIC ASSOCIATION DESCRIPTOR Command - Binary Form.....	10-3
10.2.3	GRAPHIC ASSOCIATION DESCRIPTOR Command - Interface Form.....	10-3
10.2.4	GRAPHIC ASSOCIATION DESCRIPTOR Command Restrictions.....	10-4
10.2.5	GRAPHIC ASSOCIATION DESCRIPTOR Command Examples...	10-4
10.3	CONTINUE Command.....	10-4
10.3.1	CONTINUE Command - ASCII Form.....	10-5
10.3.2	CONTINUE Command - Binary Form.....	10-5
10.3.3	CONTINUE Command - Interface Form.....	10-5
10.3.4	CONTINUE Command Restrictions.....	10-6
10.3.5	CONTINUE Command Examples.....	10-6
10.4	ASSOCIATIVE VALUES Command - Define Attribute Data.....	10-7
10.4.1	ASSOCIATIVE VALUES Command - ASCII Form.....	10-8
10.4.2	ASSOCIATIVE VALUES Command - Binary Form.....	10-9
10.4.3	ASSOCIATIVE VALUES Command - Interface Form.....	10-9
10.4.4	ASSOCIATIVE VALUES Command Restrictions.....	10-10
10.4.5	ASSOCIATIVE VALUES Command Examples.....	10-10
10.5	DRAWING IDENTIFICATION Command.....	10-11



## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
10. 5. 1	DRAWING IDENTIFICATION Command - ASCII Form.....	10-12
10. 5. 2	DRAWING IDENTIFICATION Command - Binary Form.....	10-13
10. 5. 3	DRAWING IDENTIFICATION Command - Interface Form...	10-13
10. 5. 4	DRAWING IDENTIFICATION Command Restrictions.....	10-15
10. 5. 5	DRAWING IDENTIFICATION Command Examples.....	10-15
10. 6	RANGE Command.....	10-15
10. 6. 1	RANGE Command - ASCII Form.....	10-15
10. 6. 2	RANGE Command - Binary Form.....	10-16
10. 6. 3	RANGE Command - Interface Form.....	10-16
10. 6. 4	RANGE Command Restrictions.....	10-17
10. 6. 5	RANGE Command Examples.....	10-17
10. 7	CHANGE Command.....	10-18
10. 7. 1	CHANGE Command - ASCII Form.....	10-18
10. 7. 2	CHANGE Command - Binary Form.....	10-18
10. 7. 3	CHANGE Command - Interface Form.....	10-19
10. 7. 4	CHANGE Command Restrictions.....	10-19
10. 7. 5	CHANGE Command Examples.....	10-20
11.	SIF TAPE FORMATS.....	11-1





# LIST OF FIGURES

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1-1	SIF Mediums of Exchange.....	1-2
2-1	Sample SIF Environment File.....	2-2
2-2	DEC Format for Byte Data.....	2-4
2-3	STD Format for Byte Data.....	2-5
2-4	Example of Symbol named 'XHAIR'.....	2-6
2-5	Example of SIF Symbol Masking.....	2-8
2-6	DEC Format of 32-bit Integer Data.....	2-19
2-7	STD Format of 32-bit Integer Data.....	2-20
2-8	Example of Linear Patterning.....	2-27
3-1	Example of SIF Index File Format.....	3-1
5-1	SIF Binary Header Word.....	5-1
5-2	SIF Signed Value Format.....	5-2
6-1	Example of VAX LINK Command File.....	6-1
6-2	Subroutines in the SIF Interface Libraries.....	6-7
7-1	Example of Construction Line Display.....	7-10
7-2	Example of Different Fonts.....	7-16
7-3	Example of SIF Linear Patterning.....	7-21
7-4	Example of SIF Area Patterning.....	7-22
7-5	Example of Various Line Weights.....	7-27
7-6	Example of Various Line Styles.....	7-28
7-7	Example of SIF Justifications.....	7-33
7-8	Example of SIF Paragraph.....	7-37
7-9	Example of SIF TEMPORARY ORIGIN Command.....	7-42
7-10	Database Linkage Format.....	7-43
7-11	User Linkage Format.....	7-44
8-1	Example of SIF 2-D Lines and Line Strings.....	8-6
8-2	Example of SIF 3-D Lines and Line Strings.....	8-6
8-3	Four Forms of the CIRCLE Command - Binary Form....	8-9
8-4	Example of SIF Concentric Circles.....	8-12
8-5	Examples of SIF 3-D Circles.....	8-12
8-6	Example of SIF Area Patterning.....	8-13
8-7	Forms of the ARC Command - ASCII Form.....	8-16
8-8	Example of SIF Circular Arcs.....	8-19
8-9	Example of SIF 3-D Arcs (ISO View).....	8-19
8-10	Example of SIF Arcs.....	8-20
8-11	Example of SIF Scaled Symbol.....	8-25
8-12	Example of SIF Rotated Symbol.....	8-25
8-13	Example of SIF 2-D Curves.....	8-33
8-14	Example of SIF 3-D Curves.....	8-33
8-15	Example of SIF Ellipses.....	8-38
8-16	Example of SIF 3-D Ellipses (ISO View).....	8-39
8-17	ELLIPTICAL ARC Command - Binary Form.....	8-42
8-18	Example of SIF 2-D Elliptical Arcs.....	8-45
8-19	Example of SIF 3-D Elliptical Arcs.....	8-45
8-20	Example of SIF Elliptical Arc in Complex String...	8-46



# LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Title</u>	<u>Page</u>
8-21	Example of SIF 2-D Conic.....	8-50
8-22	Example of SIF 3-D Conic.....	8-50
8-23	Example of SIF Complex String.....	8-53
8-24	Example of SIF Expanded Symbol.....	8-58
8-25	Example of SIF Created Symbol.....	8-58
8-26	Example of SIF Created Symbol.....	8-58
8-27	Example of SIF Surface.....	8-64
8-28	Example of SIF Solid.....	8-68
8-29	Example of SIF 2-D Point String.....	8-74
8-30	Example of SIF 3-D Point String.....	8-75
8-31	Example of 3-D SIF Cylinder as a Solid.....	8-81
8-32	Example of 3-D SIF Cylinder as a Solid Surface....	8-83
8-33	Example of 3-D SIF Cone as a Solid.....	8-88
8-34	Example of 3-D SIF Cone as a Surface.....	8-90
8-35	Example of 2-D B-Spline Curve (uniform, rational).....	8-94
8-36	Example of 2-D SIF B-Spline Curve (uniform, non-rational).....	8-95
8-37	Example of 2-D B-Spline Curve (non-uniform, non-rational).....	8-97
8-38	Example of 2-D B-Spline Curve (non-uniform, rational).....	8-98
8-39	Example of 3-D B-Spline Curve (non-uniform, rational).....	8-99
8-40	Example of B-Spline Surface (non-uniform, non-rational).....	8-107
8-41	Example of 3-D SIF B-Spline Surface.....	8-109
9-1	Example of SIF Text Lines at Varying Angles.....	9-7
9-2	Example of SIF Text Lines at Varying Justifications.....	9-7
9-3	Example of SIF 3-D Text Lines.....	9-9
9-4	Example of SIF Text Lines with Varying Fonts.....	9-9
9-5	Example of Text Node at 45 Degrees.....	9-17
9-6	Example of SIF Text Node at 90 Degrees.....	9-17
9-7	Example of SIF Enter Data Field Within Text Node.....	9-20
9-8	Example of SIF Text Node Using PA3/ Command.....	9-23
11-1	SIF Tape Format for ASCII and Binary Files.....	11-2



## LIST OF TABLES

<u>Figure</u>	<u>Title</u>	<u>Page</u>
7-1	IGDS Element Classes.....	7-8
7-2	IGDS Pattern Types.....	7-17
7-3	IGDS Line Styles.....	7-23
7-4	SIF Justification versus IGDS Justification.....	7-30



## 1. INTRODUCTION

The Standard Interchange Format (SIF) is a generic format for transmittal of graphic data and associated nongraphic data between systems. This document describes the implementation of SIF on the Intergraph system. The medium of exchange for SIF files between systems is magnetic tape as presented in Figure 1-1. The SIF tape format is described in Section 11. The control file for defining SIF file characteristics is the SIF environment file and is described in Section 2.

Each SIF command has both an ASCII and a binary form. The ASCII form is more readable but requires extra processing time for parsing the commands. The SIF ASCII and binary command formats are presented in Sections 7 through 10. An interface form is also available for the FORTRAN users creating SIF files from an application's task. The use of the SIF interface libraries is described in Section 6. When examples of SIF commands are given, the ASCII form is used due to readability.

### 1.1 SIF Processors

A brief description of each SIF processor is presented in this section. This description is necessary since the SIF processors are referenced by name in subsequent sections. These processors are provided with the Intergraph SIF product and are not necessarily available on non-Intergraph systems which support the SIF format for graphic file exchange.

The APT processor is provided to aid the user when the patterning features are used. When patterns are defined during SIF-in processing, only the element to be patterned and the pattern description are written to the output design file. The IGDS pattern driver (PDV) processor must be executed as a postprocessor to ATQ or TRI to generate the pattern component elements. APT creates a file which contains the command line to execute the PDV processor. In single file processing mode, the file contains one command line. In multiple file processing mode, the file contains a command line for each design file generated during SIF-in processing.

The ASI processor is provided to translate a SIF ASCII file on disk to a SIF binary file on disk. ASI automatically generates SIF binary CONTINUE commands for those commands which may be continued. If executed with a SIF index file, ASI also creates an output SIF binary index file.

The ASD processor is provided to translate a SIF binary file on disk to a SIF ASCII file on disk. You can specify the maximum length of an ASCII file record in the SIF environment file. If executed with a SIF index file, ASD also creates an output SIF ASCII index file.

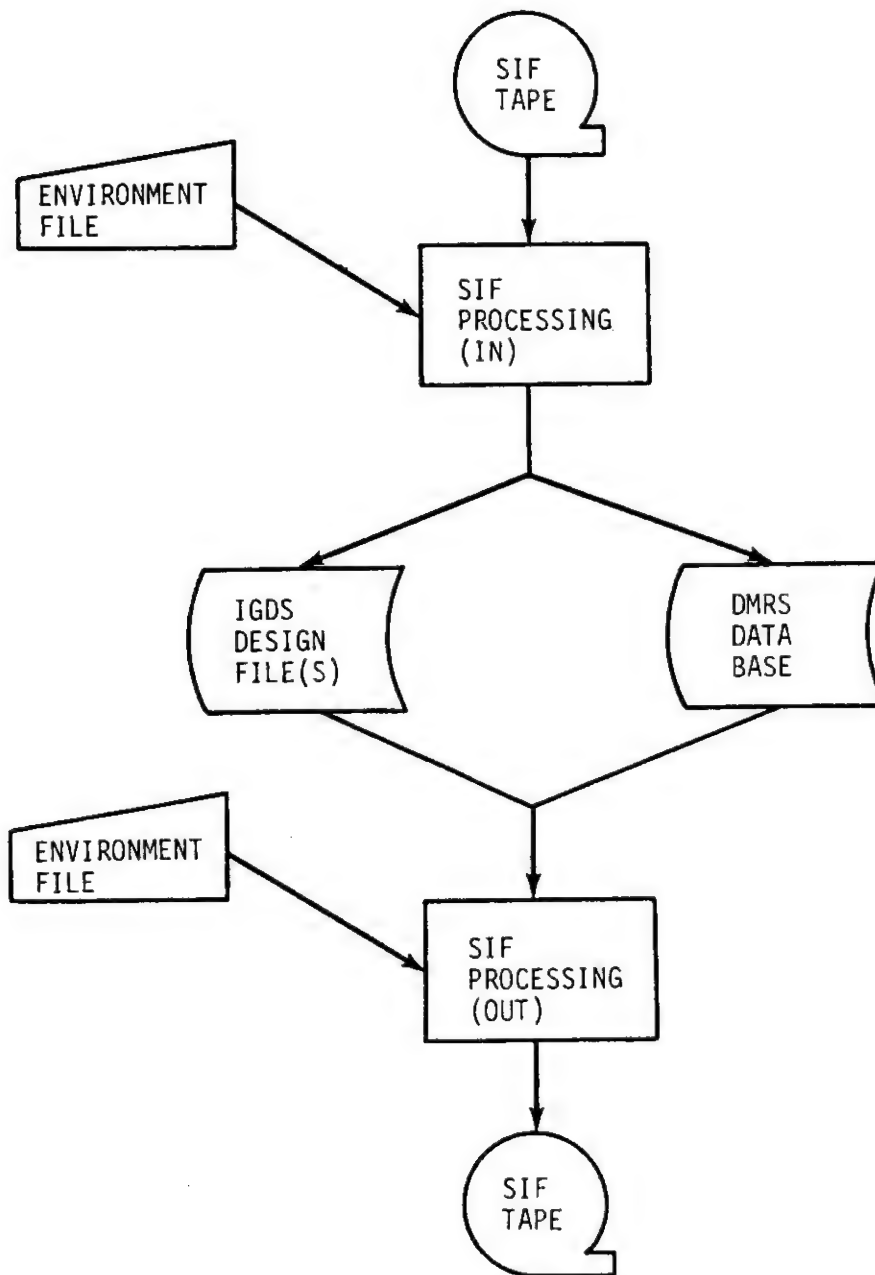


Figure 1-1. SIF Mediums of Exchange



The ATQ processor is provided to translate a SIF ASCII file into an IGDS design file and associated DMRS database. The design file is always created during SIF-in processing. The database must exist before SIF-in processing. If executed with a SIF index file, ATQ also creates an output SIF design index file.

The ATI processor is provided to translate a SIF ASCII file on tape to a SIF ASCII file on disk. The tape must be in the format specified in Section 10. ATI also processes tapes which are in EBCDIC format instead of ASCII. You can translate a single file or multiple files on the tape. If multiple files are processed, you can specify the SIF ASCII index file to be used or request that ATI create a SIF ASCII index file of all files processed.

The ATO processor is provided to translate a SIF ASCII file on disk to a SIF ASCII file on tape. ATO also creates tapes which are in EBCDIC format instead of ASCII. You can translate a single file or multiple files to tape.

The BTI processor is provided to translate a SIF binary file on tape to a SIF binary file on disk. The tape must be in the format specified in Section 10. You can translate a single file or multiple files on the tape. If multiple files are processed, you can specify the SIF binary index file to be used or request that BTI create a SIF binary index file of all files processed.

The BTO processor is provided to translate a SIF binary file on disk to a SIF binary file on tape. You can translate a single file or multiple files to tape.

The CPC processor is provided to create IGDS cells from elements placed in the design file during ATQ or TRI processing. These cells are placed in the IGDS cell library specified in the SIF environment file. The cell library is created if it does not already exist.

The DLT processor is provided to delete entities which were inserted during a previous execution of ATQ or TRI. When ATQ or TRI inserts an entity into the DMRS database, the occurrence number of the entity is also written to a file which is named using the SIF file name and an extension of AS. If a problem is detected after SIF-in processing, you can execute the DLT processor to delete only the entities inserted by ATQ or TRI.

The FENCE processor is provided to support the FENCE user command which allows you to select a specific area of the design file for SIF-out processing. The user command allows you to place a fence while viewing the design file. The user command only updates the specified SIF environment file with the fence coordinates in UORs. The SIF-out processor must be executed separately.

The GTA processor is provided to translate an IGDS design file and associated DMRS database into a SIF ASCII file. If executed with a SIF index file, GTA also creates an output SIF ASCII index file.

The HLP processor is provided to aid you in analyzing the contents of the SIF binary file. You are allowed to define certain search criteria, such as SIF command types and the starting and ending commands. You can request output to the terminal or to a disk file. You can also write specified binary commands to an output binary file to allow processing of a subset of the entire binary file.

The NDX processor is provided to allow you to create, restore, or edit a SIF index file. You must answer prompts to indicate the processing mode. If a SIF index file needs to be edited, NDX must be used instead of an on-line edit process, which destroys the direct access characteristics of the index file. The command procedure SIFNDX is also provided to allow you to create a SIF index file without having to enter the file specification for each file to be included in the index file.

The RCV processor is provided to allow you to recover from certain errors encountered during SIF-in processing without having to execute the SIF-in processor again. You may currently recover only from database errors in attribute specifications. The SIF-in processor writes a recovery file using the SIF file name and an extension of RCV. This recovery file can be edited with an on-line edit process before executing RCV.

The SIFERR processor is provided to list the SIF error message for the given SIF error number. SIFERR recognizes one or more error numbers on the command line or prompts for error numbers if no parameters were entered on the command line.

The TRI processor is provided to translate a SIF binary file into an IGDS design file and associated DMRS database. The design file is always created during SIF-in processing. The database must exist before SIF-in processing. If executed with a SIF index file, TRI also creates an output SIF design index file.

The TRO processor is provided to translate an IGDS design file and associated DMRS database into a SIF binary file. If executed with a SIF index file, TRO also creates an output SIF binary index file.

The TRU processor is provided to truncate the design files created during SIF-in processing. There is an option which allows the SIF-in processor to truncate the design files. However, the SIF-in processor should not truncate the design file if patterns were specified or if cells must be created by the CPC processor.

## NOTES

## 2. SIF ENVIRONMENT FILE

The SIF environment file is the control file required by all SIF processors except NDX and SIFERR. The environment file contains file specifications and characteristics required for SIF processing. The environment file is a formatted, sequential file with variable length records. The environment file can be created with an on-line process which creates a file with the proper file characteristics. The command procedure SIFENV is delivered with the SIF product to aid in creating the environment file. SIFENV prompts you for the information required for SIF processing and writes the environment file. SIFENV can be executed with one of the following command lines:

```
$ SIFENV
```

```
$ @PRO_DD_SIF: SIFENV
```

Environment file records consist of a two-character keyword followed by an equal sign "=" followed by a data specification. Each keyword specification must appear on a separate record. The keywords may appear in any order in the environment file. If the default value is required or the SIF processor does not use the keyword, the keyword specification may be omitted from the environment file or included by using a dollar sign "\$" for the data specification. A sample environment file using all keywords recognized by SIF processors is presented in Figure 2-1. The following sections describe each keyword recognized by SIF processors.

### 2.1 AR - Define the ARC Command Format for SIF-out

The AR keyword allows you to specify which form of the ARC command should be created during SIF-out processing. The AR keyword is recognized by the GTA and TRD processors. Two forms of the ARC command are available. The edge form allows a circular arc to be defined by three vertices on the arc. The origin form allows a circular arc to be defined by the origin, two vertices on the arc, and a sweep direction. If the AR keyword is not specified in the environment file, the origin form of the arc is created. The format of the AR keyword follows:

```
AR=EDGE
```

```
AR=ORIGIN
```

AR=ORIGIN  
AS-S F ASC  
BY-DEC  
CL-S F CEL  
CO=2  
CZ=0  
DB-S F DBS  
DE=0  
DF-S F DON  
DM=N  
DP=N  
DU=4294967295.011FT IN  
EO=Y  
EL=63  
EN-SYS OUTPUT  
ER=N  
ET=N  
FE=0  
FS=500/TRG3  
FT=Y  
IN=DEC  
IX=0  
LR=64  
MA=N  
MO=2  
NE=Y  
OV=ALL  
PF=0  
PT=Y  
RA=N  
SD=SEED DON  
SX-S F BIN  
TC=N  
TD=1600  
TF=ALL  
TN=ORIGIN  
TX=ORIGIN  
TY=ALL

Figure 2-1. Sample SIF Environment File

A keyword specification of STD indicates that byte data on the SIF binary tape is in a format which stores byte data left/right in a 16-bit word. The characters ABCD in STD format are presented in Figure 2-3. When transferring SIF binary tape files between systems where both systems use the STD format for byte storage, the bytes in the SIF binary commands do not have to be modified. When transferring from a system with STD format to a system with DEC format, some modification of the binary commands is necessary before the SIF file can be processed on the receiving system. The bytes in each 16-bit word of the binary commands must be swapped. This byte swap applies to the fields of all binary commands except those commands which have ASCII fields (pattern, symbol, text, etc.). The ASCII fields in binary commands must not be swapped.

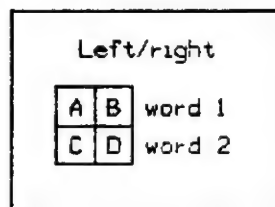


Figure 2-3. STD Format for Byte Data

#### 2.4 CL - Define the IGDS Cell Library File

The CL keyword allows you to define the IGDS cell library file specification to be used during SIF processing. The CL keyword is recognized by the ATG, CPC, GTA, TRI, and TRO processors. The recommended extension for the cell library file is .CEL. The cell library contains the graphic definition for all SYMBOL commands in the SIF file. The cell library must be a contiguous file. If the CL keyword is omitted, all SYMBOL commands are tagged as orphan cells. The format of the CL keyword follows:

### 2.3 BY - Define the Byte Storage for SIF Binary Tapes

The BY keyword allows you to specify the format for byte storage on the SIF binary tape. The BY keyword is recognized by the BTI and BTD processors. If the BY keyword is not specified in the environment file, the default specification of DEC is used. The format of the BY keyword follows:

BY=DEC

BY=STD

A keyword specification of DEC indicates that byte data on the SIF binary tape is in a format which stores byte data right/left in a 16-bit word. The characters ABCD in DEC format are presented in Figure 2-2. When transferring SIF binary tape files between systems where both systems use the DEC format for byte storage, the bytes in the SIF binary commands do not have to be modified. When transferring from a system with DEC format to a system with STD format, some modification of the binary commands is necessary before the SIF file can be processed on the receiving system. The bytes in each 16-bit word of the binary commands must be swapped. This byte swap applies to the fields of all binary commands except those commands which have ASCII fields (pattern, symbol, text, etc.). The ASCII fields in binary commands must not be swapped.

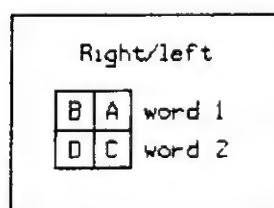


Figure 2-2. DEC Format for Byte Data



The /MK switch applies only to SIF-out processing. This switch allows special processing of cells which have been masked with the MASK user command in the graphics support product. MASK allows the user to delete a portion or portions of a cell and retain some cell intelligence. The cell is changed to an orphan cell, and all attribute linkage is retained. A graphics support user linkage is added to the orphan cell header to indicate that the cell has been masked. MASK is especially useful for mapping applications where two or more cells overlap and do not appear reasonable when plotted. If the /MK switch is specified, a SIF symbol set is created, but only the mask components are included after the SYMBOL command. An example of cell masking is presented in Figure 2-5. The cell named CONCIR consists of four concentric circles. The cell named CONDIA consists of four concentric diamonds. The left section of the figure is the graphic picture of the overlapping cells after they are placed in the graphics file. The middle section of the figure shows the cell CONCIR after cell masking. The dotted triangle is the portion of the cell which was masked. The cell CONDIA is placed on a different level and is not displayed in the middle figure. The right section of the figure shows the final graphic picture after cell masking. The dotted triangle is construction class, and construction line display has been turned off. The SIF symbol set which is created for these cells during SIF-out processing follows. The cell named CONDIA was not cut and is not expanded. Only the cell named CONCIR which was cut causes a SIF symbol set to be created.

```

SYM/OR=15000,3000,MA=5.,0.,0.,5.,CONDIA
BSY/MK
SYM/OR=15000,5000,MA=5.,0.,0.,5.,CONCIR
CLA/2
LAC/LT=1
LAC/LS=1
LAC/LC=1
LST/MK,14999,5002,16001,2998,13984,2990,14999,5002
ESY/

```

CL=cellfile

CL=cellfile/EX

CL=cellfile/MK

CL=cellfile/EF=excfile

CL=cellfile/EX/EF=excfile

CL=cellfile/CR

CL=cellfile/FS=blocks

The cellfile field defines the file specification for the IGDS cell library. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

The /EX switch applies only to SIF-out processing. If the /EX switch is not specified, only a SYMBOL command is created for each cell processed. If the /EX switch is specified, a SIF symbol set is created for each cell processed. A SIF symbol set consists of a BEGIN SYMBOL command, SYMBOL command, commands to define all cell components, and an END SYMBOL command. An IGDS cell named XHAIR which consists of a circle and two line elements is presented in Figure 2-4. The SIF symbol set which is created for this cell during SIF-out processing follows:

```
BSY/  
SYM/OR=1000,1000,MA=1.,0.,0.,1.,XHAIR  
CIR/RA=500,CE=1000,1000  
LST/0,1000,2000,1000  
LST/1000,0,1000,2000  
ESY/
```

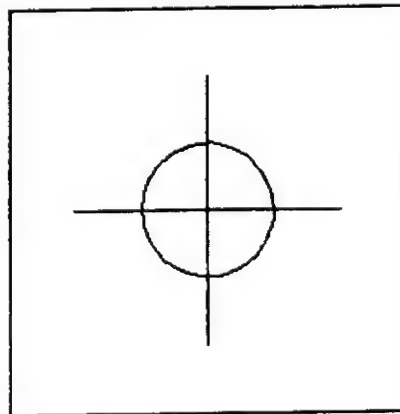


Figure 2-4. Example of Symbol named 'XHAIR'

fictitious, the CPC processor creates a new cell library using the fictitious name and defines the cell in the new cell library using the component elements found in the original expanded symbol set. The /FS switch applies only when the CPC processor is creating a new cell library. The /FS switch indicates, in blocks, the size of the new cell library to be created. If this switch is not specified, the default value is the value indicated in the FS keyword in the environment file.

The /CR switch applies to SIF-out processors only. When this switch is used, the first occurrence of each cell encountered in the IGDS design file is expanded into a SIF symbol set with the Create option on the SIF ASCII BEGIN SYMBOL command (BSY/CR). A SIF SYMBOL command immediately follows the symbol creation set. It should be noted that the /CR switch and /EX switch should not be used concurrently.

## 2.5 CO - Define the Integer Representation for SIF Binary Tapes

The CO keyword allows you to specify the representation of integer data on the SIF binary tape. The CO keyword is not implemented in the current release of SIF. When implemented, the CO keyword is recognized by the BTI and BTO processors. The current release assumes that all integers are stored in two's complement representation.

## 2.6 CZ - Define the Constant Z for 2-D to 3-D SIF Translations

The CZ keyword allows you to specify the constant Z coordinate for 2-D to 3-D SIF translations. The CZ keyword is not implemented in the current release of SIF. When implemented, the CZ keyword is recognized by the ATG, GTA, TRI, and TRO processors. The current release translates files from 2-D to 2-D and from 3-D to 3-D.

## 2.7 DB - Define the DMRS Database File Name

The DB keyword allows you to define the DMRS database file specification to be used during SIF processing. The DB keyword is recognized by the ATG, DLT, GTA, REC, TRI, and TRO processors. The recommended extension for the database is DBS. The database must exist and be a contiguous file. If the DB keyword is omitted, all commands which pertain to database processing are ignored. The format of the DB keyword follows:

DB=database

DB=database/NA=dbname

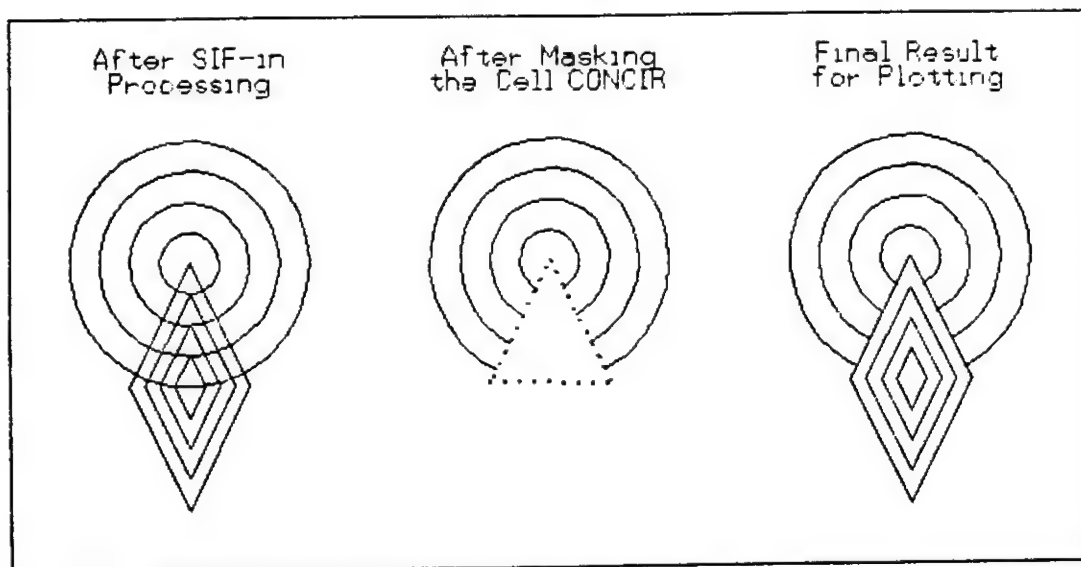


Figure 2-5. Example of SIF Symbol Masking

The /EF switch applies only to SIF-out processing and allows you to specify a cell exceptions file. The excfile field defines the file specification for the exceptions file. The status of the /EX switch determines how the exceptions file is interpreted. If the /EX switch is not specified, the exceptions file is assumed to contain a list of cell names which should be expanded. If the /EX switch is specified, the exceptions file is assumed to contain a list of cell names which should not be expanded.

The exceptions file is a formatted, sequential file which may be created with any on-line edit process. Each record in the file may contain only one cell name of 1 to 6 characters. The cell names may appear in any order in the file. The maximum number of cell names which may appear in the exceptions file is 125.

If the cell library specified does not exist or the default specification (CL=) appears in the environment file, all symbols that appear in the SIF ASCII file are created as SIF orphan cells. The design file containing the SIF orphan cells can then be used as input to the CPC processor. By changing the cell library specification in the environment file to an existing cell library or to a fictitious cell library, the CPC processor redefines the SIF orphan cell in the design file using the definition in the specified cell library. If the cell library name specified in the environment file is

During SIF-out processing, the DF keyword can contain the file specification of a design file or cell library. The design file or cell library must already exist and be a contiguous file. When a cell library is processed, the cells are automatically expanded and the Create option is specified on the SIF ASCII BEGIN SYMBOL command (BSY/CR).

## 2.10 DM - Define the Print Attribute Data Option

The DM keyword allows you to specify the status of the Print Attribute Data option. The DM keyword is recognized by the ATG and TRI processors. The attribute values for each entity inserted during SIF processing are defined in the ASSOCIATIVE VALUES command. If the DM keyword is not specified in the environment file, attribute data is not printed. The format of the DM keyword follows:

DM=N

DM=Y

A keyword specification of N indicates that all attribute data should not be printed to the message file. The only case where attribute data is printed to the message file is for commands in which an error is detected.

A keyword specification of Y indicates that all attribute data should be printed to the message file whether an error is detected or not. This option allows you to keep a record of all information inserted into the database for each SIF translation.

## 2.11 DP - Define the Drop Paragraph Option

The DP keyword allows you to specify the status of the drop paragraph option. The DP keyword is recognized by the GTA and TRD processors. This keyword allows you to specify how text nodes are processed during SIF-out. If the DP keyword is not specified in the environment file, a SIF paragraph is created for each text node processed. The format of the DP keyword follows:

DP=N

DP=Y

The database field defines the file specification for the DMRS database. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

The /NA switch allows you to specify the database name if the database was assigned a name when created. The dbname field defines the database name and may be from 1 to 30 characters consisting of A-Z, 0-9, and underbar. If specified, the database name must match the specification in the DDL source file. If not specified, the database name is not checked. Therefore, the database name does not have to be specified for SIF to attach it for processing.

## 2.8 DE - Define the Debug Processing Option

The DE keyword allows you to receive debug messages in the SIF message file. The DE keyword is not implemented in the current release of SIF. When implemented, the DE keyword is recognized by all processors. The current release does not print any debug messages to the message file.

## 2.9 DF - Define the IGDS Design File Name

The DF keyword allows you to define the IGDS design file specification to be used during SIF processing. The DF keyword is recognized by the APT, ATG, DLT, GTA, RCV, TRI, TRD, and TRU processors. The recommended extension for the design file name is .DGN. The format of the DF keyword follows:

DF=filespec

The filespec field defines the file specification for the IGDS design file. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

The IGDS design file is one of the input files to the GTA, ASI, TRD, and TRU processors and, therefore, must exist before these processors can be executed. The IGDS design file is one of the output files created by the ATG and TRI processors. The APT and TRU processors only use the IGDS design file specification to form other file specifications required during processing.

units (UDRs) per subunit. Any one of these fields can be omitted, which causes the SIF processor to calculate the maximum value for that field. The design file has a range of 0 to 4294967295 along each axis. The product of the fields defining the working units must be less than or equal to 4294967295.

The mudesc field is a two-character description of the master units field (FT for feet, M for meters, etc.). The sudesc field is a two-character description of the subunits field (IN for inches, CM for centimeters, etc.).

## 2.13 ED - Define the Process Enter Data Field Option

The ED keyword allows you to specify the status of the process Enter Data Field option. The ED keyword is recognized by the GTA and TRD processors. This keyword allows you to specify how enter data fields should be processed during SIF-out. If the ED keyword is not specified in the environment file, the enter data field status is retained for all text processed. The format of the ED keyword follows:

ED=N

ED=Y

A keyword specification of N indicates that the enter data field status should be dropped. All enter data fields are processed as if the enter data field text were part of the text string. The only exception is when cells are not expanded during SIF-out. In this case, the SYMBOL command created is followed by one or more INCLUDE TEXT commands. The following example contains a TEXT command and a PARAGRAPH command set after SIF-out processing where enter data fields are dropped. The text element in the design file contains 15 characters. Each line in the text node contains 15 characters. In both the text and text node elements, the numeric characters form the enter data fields, and the alpha characters are normal text.

```
TLC/CO=15, JU=3
TXT/TH=100, TW=100, OR=0, 0, AN=0, AAA111BBB222CCC
TPC/CO=15, NU=3, JU=1, SP=50
PAR/FO=0, ID=0, TH=100, TW=100, OR=1000, 1000, AN=0
PLN/AAA111BBB222CCC
PLN/DDD333EEE444FFF
PLN/GGG555HHH666III
CLP/
```



A keyword specification of N indicates that the paragraph status should not be dropped. A SIF PARAGRAPH command set should be created for each text node processed. An example of a SIF PARAGRAPH command set follows:

```
TPC/CO=11, NU=6, JU=5, SP=50
PAR/FO=0, ID=0, TH=100, TW=100, OR=0, 0, AN=0
PLN/This is a
PLN/sample of a
PLN/paragraph
PLN/which is
PLN/center
PLN/justified.
CLP/
```

A keyword specification of Y indicates that the paragraph status should be dropped. Each text node processed is translated into a SIF TEXT command for each line in the text node. Even though the paragraph status is dropped, the resulting graphic picture of the text node is retained. An example of dropping the paragraph status follows using the same text node in the preceding example.

```
TLC/CO=9, JU=5
TXT/TH=100, TW=100, OR=0, 375, AN=0, This is a
TLC/CO=11, JU=5
TXT/TH=100, TW=100, OR=0, 225, AN=0, sample of a
TLC/CO=9, JU=5
TXT/TH=100, TW=100, OR=0, 75, AN=0, paragraph
TLC/CO=8, JU=5
TXT/TH=100, TW=100, OR=0, -75, AN=0, which is
TLC/CO=6, JU=5
TXT/TH=100, TW=100, OR=0, -225, AN=0, center
TLC/CO=10, JU=5
TXT/TH=100, TW=100, OR=0, -375, AN=0, justified.
```

## 2.12 DU - Define the Design File Working Units

The DU keyword allows you to define the working units for the design files created during SIF processing. The DU keyword is recognized by the ATG and TRI processors. The format of the DU keyword follows. If the DU keyword is not specified in the environment file, the working units defined in the seed file are used.

```
DU=mudf:sumu:pusu:mudesc:sudesc
```

The mudf field defines the number of master units in the design file. The sumu field defines the number of subunits per master unit. The pusu field defines the number of positional



EM=filspec/FU

EM=SYS\$OUTPUT/FU

EM=/FU

The filespec field defines the file specification of the SIF message file. If specified, the same file specification is used by all SIF processors. A new version of the message file is created for each execution of a SIF processor. If the file specification is SYS\$OUTPUT, all messages are written to the user's terminal.

The /FU switch allows you to receive SIF error messages as well as SIF error numbers for each error detected during SIF processing. The messages are identical to the messages given by the SIFERR processor. If the /FU switch is not specified, only the SIF error number is written to the message file.

## 2.16 ER - Define the Recovery Option

The ER keyword allows you to specify the status of the Recovery option. The ER keyword is recognized by all processors except FENCE, HLP, NDX, and SIFERR. If the ER keyword is not specified in the environment file, processing terminates when the first error is detected. The format of the ER keyword follows:

ER=N

ER=Y

A keyword specification of N indicates that processing should terminate when the first error is detected. If you are processing with a SIF index file, processing terminates on the current SIF file, and the next SIF file in the index file is processed.

A keyword specification of Y indicates that the SIF processor should continue if a recoverable error is detected. The commands which are in error are not processed, but all commands can be processed in order to detect more than one error during each execution. The SIF error numbers which are considered to be recoverable have a range of 8000 to 8999. Error numbers in the range of 9000 to 9999 are considered to be fatal and cause processing to terminate.

A keyword specification of Y indicates that the enter data field status should not be dropped. All enter data fields are retained with INCLUDE TEXT commands after the SYMBOL and TEXT commands. If cells are expanded, the INCLUDE TEXT commands appear after the END SYMBOL command. Enter data fields in paragraph lines are preserved with the edit delimiter in the PARAGRAPH LINE command. The following example contains a SIF TEXT command and a PARAGRAPH command set after SIF-out processing where enter data fields are retained. The text element in the design file contains 15 characters. Each line in the text node contains 15 characters. In both the text and text node elements, the numeric characters form the enter data fields, and the alpha characters are normal text.

```
TLC/CO=15, JU=3
TXT/TH=100, TW=100, OR=0, 0, AN=0, AAA___BBB___CCC
INC/LI=1, DE=!, !111!222!
TPC/CO=15, NU=3, JU=1, SP=50
PAR/FO=0, ID=0, TH=100, TW=100, OR=1000, 1000, AN=0
PLN/NE=!, ED=", !AAA!"111"!BBB!"222"!CCC!
PLN/NE=!, ED=", !DDD!"333"!EEE!"444"!FFF!
PLN/NE=!, ED=", !GGG!"555"!HHH!"666"!III!
CLP/
```

#### 2.14 EL - Define the Error Level

The EL keyword allows you to define the level which should contain all error graphics. The EL keyword is recognized by the ATG and TRI processors. If the EL keyword is not specified in the environment file, the default for error graphics is level 63. The format of the EL keyword follows:

EL=level

The level field defines the error level and can range from 1 to 63. Any value outside this range is changed to 63. During SIF processing, graphics for any commands which are defined on a level outside the range of 1 to 63 are placed on the error level.

#### 2.15 EM - Define the SIF Message File

The EM keyword allows you to define the SIF message file specification. The EM keyword is recognized by all SIF processors except FENCE, HLP, NDX, and SIFERR. The recommended extension for the message file is MSG. If the EM keyword is not specified in the environment file, the message file is named using the SIF processor name for the file name and an extension of MSG (ASI.MSG for the ASI processor, etc.). The format of the EM keyword follows:

## 2.19 FL - Define the IGDS Font Library File

The FL keyword allows you to define the IGDS font library file specification to be used during SIF processing. The FL keyword is recognized by the ATG, TRI, GTA, and TRD processors. The font library contains any special fonts you may wish to access during text, text node, and symbol processing. If the FL keyword is omitted, all font processing accesses. The format of the FL keyword is as follows:

FL=font library

The font library field defines the file specification for the IGDS font library. The device, UIC, file name, extension, and version may be specified. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version number if not specified.

## 2.20 FS - Define the IGDS Design File Size

The FS keyword allows you to define the file size in blocks of the design file created during SIF-in processing. The FS keyword is recognized by the ATG, TRI, and TRU processors. If the FS keyword is not specified in the environment file, a design file of 500 blocks is allocated. The format of the FS keyword follows:

FS=size/TR:percent

FS=size/-TR:percent

The size field defines the number of contiguous blocks to be initially allocated for the design file. If this field is not specified, a file of 500 blocks is allocated.

The /TR and /-TR switches indicate the status of the Truncate Design File option. The /TR switch indicates that the design file should be truncated after ATG or TRI processing. The /-TR switch indicates that the file should not be truncated after ATG or TRI processing. The TRU processor truncates the design file regardless of the /-TR switch.

The percent field defines the percentage of free space that you desire. The percentage has a range of 0 to 100 and defaults to 33 if not specified. The truncated file size is derived by starting with the number of blocks actually used, adding percent of the blocks actually used, and rounding to the next tenth block. The truncated file size cannot be larger than the original allocation.

## 2.17 ET - Define the EBCDIC Tape Format Option

The ET keyword allows you to process a SIF tape in EBCDIC format. The ET keyword is recognized by the ATI and ATO processors. If the ET keyword is not specified in the environment file, the tape is assumed to be in ASCII format. The format of the ET keyword follows:

ET=N

ET=Y

A keyword specification of N indicates that the tape is not in EBCDIC format. All tape files processed are assumed to be in ASCII format.

A keyword specification of Y indicates that the tape is in EBCDIC format. During ATI processing, all commands on tape are converted from EBCDIC to ASCII before being written to disk. During ATO processing, all commands on disk are converted from ASCII to EBCDIC before being written to tape.

## 2.18 FE - Define the Fence for SIF-out Processing

The FE keyword allows you to define the fence to be used during SIF-out processing. The FE keyword is recognized by the GTA and TRO processors. This keyword allows only elements in a specified area of the design file to be processed. The element is processed if any portion of the element lies inside the fence and no clipping is performed. If the FE keyword is not specified in the environment file, all elements in the design file are processed. The format of the FE keyword follows:

FE=xlow,ylow,xhigh,yhigh

FE=xlow,ylow,zlow,xhigh,yhigh,zhigh

The xlow,ylow,zlow field defines the lowest X, Y, and Z coordinates for the fence in UORs. The xhigh,yhigh,zhigh field defines the highest X, Y, and Z coordinates for the fence in UORs. The Z coordinates must not be specified for 2-D processing but are required for 3-D processing.

The user command FENCE is delivered with the SIF product and allows you to select the fence easily while viewing the design file. FENCE works only with 2-D design files. When FENCE is executed, you must enter the SIF environment file to be updated. Then you must enter the lower left and upper right data points to define the fence. A new version of the specified environment file is written containing the FE keyword which describes the fence.

file can be processed on the receiving system. The 32-bit integers in each 32-bit word of the binary commands must be swapped. This swap applies to the fields of all binary commands except the command header and ASCII fields.

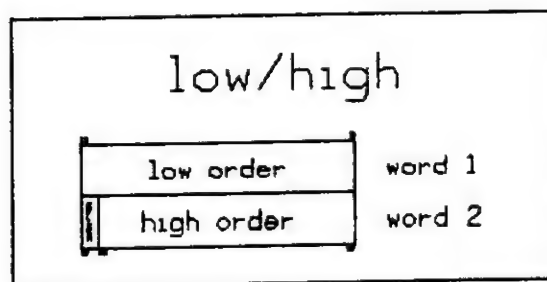


Figure 2-6. DEC Format of 32-bit Integer Data

A keyword specification of STD indicates that 32-bit integer data on the SIF binary tape is in a format which stores 32-bit integer data high/low in a 32-bit word. The STD format is presented in Figure 2-7. When transferring SIF binary tape files between systems where both systems use the STD format for 32-bit integer storage, the 32-bit integers in the SIF binary commands do not have to be modified. When transferring from a system with STD format to a system with DEC format, some modification of the binary commands is necessary before the SIF file can be processed on the receiving system. The 32-bit integers in each 32-bit word of the binary commands must be swapped. This swap applies to the fields of all binary commands except the command header and ASCII fields.

## 2.21 FT - Define the Font Processing Option

The FT keyword allows you to specify the status of the Font Processing option. The FT keyword is recognized by the ATG, GTA, TRI, and TRO processors. If the FT keyword is not specified in the environment file, the fonts specified in the font commands are used. The format of the FT keyword follows:

FT=N

FT=Y

A keyword specification of N indicates that all TEXT and PARAGRAPH commands should be created with font 0 and any font commands in the SIF file should be ignored. This allows SIF files to be translated on systems where the requested fonts are not available and prevents the font not defined error message from being written to the message file for each TEXT and PARAGRAPH command processed.

A keyword specification of Y indicates that all TEXT and PARAGRAPH commands should be processed according to font commands in the SIF file. An error message is written to the message file for each TEXT or PARAGRAPH command processed where the requested font is not available.

## 2.22 IN - Define the 32-bit Integer Storage for SIF Binary Tapes

The IN keyword allows you to specify the format for 32-bit integer storage on the SIF binary tape. The IN keyword is recognized by the BTI and BTD processors. If the IN keyword is not specified in the environment file, the default specification of DEC is used. The format of the BY keyword follows:

IN=DEC

IN=STD

A keyword specification of DEC indicates that 32-bit integer data on the SIF binary tape is in a format which stores 32-bit integer data low/high in a 32-bit word. The DEC format is presented in Figure 2-6. When transferring SIF binary tape files between systems where both systems use the DEC format for 32-bit integer storage, the 32-bit integers in the SIF binary commands do not have to be modified. When transferring from a system with DEC format to a system with STD format, some modification of the binary commands is necessary before the SIF

For each design file generated by SIF, an IGDS shape and text elements are added to the map index file. The vertices for the shape are the minimum and maximum coordinates used during the SIF translation. The text element contains 54 characters and three enter data fields. The text size varies according to the size of the shape. The shape and text elements are combined into a graphic group and assigned the next available graphic group number. These elements are placed on a level determined by the first character of the design file name. If the first character is A-Z, the elements are placed on levels 1-26, respectively. If the first character is 0-9, the elements are placed on levels 30-39, respectively. The first enter data field is contained in positions 1-20 in the text string and is not used by SIF. The second enter data field is contained in positions 22-23 and may vary according to user specification. The third enter data field is contained in positions 25-54 and is filled by SIF with the full file specification of the design file.

The /m-n field indicates which characters from the design file name should be written to the second enter data field. The m field defines the starting character position, and the n field defines the ending character position. For example, the field /5-6 and a design file name of 1234AB789 would cause the characters AB to be written in the second enter data field. If only one character is required, the field may be specified as /m. If the entire field is not specified, the first character in the design file name is used. If two characters are required, they must be consecutive characters in the design file name.

## 2.24 LR - Define the SIF ASCII Logical Record Length

The LR keyword allows you to define the logical record length in bytes of the SIF ASCII file. The LR keyword is recognized by the ASO, ATI, and ATO processors. If the LR keyword is not specified in the environment file, the default of 72 bytes is used. The format of the LR keyword follows:

LR=length

The length field defines the logical record length in bytes and can range from 20 to 80. Any value outside this range is changed to 72.



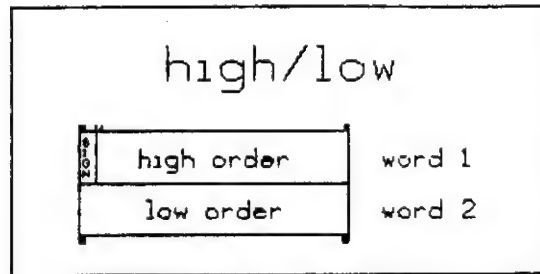


Figure 2-7. STD Format of 32-bit Integer Data

## 2.23 IX - Define the IGDS Map Index File Name

The IX keyword allows you to define the IGDS map index file specification to be used during SIF processing. The IX keyword is recognized by the ATG and TRI processors. The recommended extension for the map index file is DGN. The map index file contains information required by the Distributed Graphics Software (DGS) package. If the IX keyword is not specified in the environment file, no map index file processing occurs. The format of the IX keyword follows:

`IX=mapindex/FS=size/m-n`

The map index field defines the file specification for the IGDS map index file. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified. If the specified file does not exist, SIF allocates the file and initializes it with the first four blocks of the seed file specified with the SD keyword.

The /FS switch allows you to specify the number of blocks to allocate if the index file does not exist. This size specification is ignored if the file already exists. The size field defines the number of blocks to allocate. If the /FS switch is not specified, a file of 500 blocks is allocated.



## 2.26 MO - Define the Graphic Mode

The MO keyword allows you to define the graphic mode to be used during SIF processing. The MO keyword is recognized by all processors except NDX and SIFERR. If the MO keyword is not specified in the environment file, the default is 2-D processing. The format of the MO keyword follows:

MO=gmode

During SIF-in processing, the graphic mode specified with the MO keyword must match the graphic mode of the seed file specified with the SD keyword or an error results.

During SIF-out processing, the graphic mode specified with the MO keyword must match the graphic mode of the design file specified with the DF keyword or an error results.

## 2.27 MT - Define the Tape Drive

The MT keyword allows you to specify the tape drive used during SIF tape processing. The MT keyword applies to the SIF processors ATI, ATO, BTI and BTO. If the keyword is omitted, the default tape drive is MTAO. The format of the MT keyword follows:

MT=tapedrive

## 2.28 NE - Define the Negative Coordinate Option

The NE keyword allows you to specify whether negative coordinates should be allowed during SIF processing. The NE keyword is recognized by the ASI, ASD, ATQ, ATI, ATO, BTI, BTO, FENCE, GTA, HLP, TRI, and TRO processors. You can indicate that coordinates be both positive or negative or that all coordinates must be positive. If the NE keyword is not specified in the environment file, both positive and negative coordinates are allowed. The format of the NE keyword follows:

NE=N

NE=Y

A keyword specification of N indicates that negative coordinates are not allowed. The coordinates in the SIF file are assumed to range from 0 to 4294967295. Any negative coordinates appearing in the SIF ASCII file cause errors. The

## 2.25 MA - Define the Generate Matrix Option

The MA keyword allows you to specify whether the SIF-out processor should include a matrix in the output commands. The MA keyword is recognized by the GTA and TRD processors. If the MA keyword is not specified in the environment file, no matrix is included. The format of the MA keyword follows:

MA=N

MA=Y

A keyword specification of N indicates that the SIF command should be created without a transformation matrix. The SIF SYMBOL command always has a matrix if the symbol is placed with any rotation or scale. During 3-D processing, the matrix form of the command is always created. The following example contains SIF commands after SIF-out processing where the matrix form of the command is not generated.

```
CIR/RA=500, CE=0, 0
ARC/CC, CE=1000, 1000, P1=1500, 1000, P2=1000, 1500
ELL/CE=2000, 2000, P1=2500, 2000, P2=2000, 2250
EAR/CE=3000, 3000, P1=3500, 3000, P2=3000, 3250, ST=90, SW=180
TLC/CD=4, JU=3
TXT/TH=100, TW=100, DR=4000, 4000, AN=0, TEXT
TPC/CD=9, NU=1, JU=1, SP=50
PAR/FO=0, ID=0, TH=100, TW=100, DR=5000, 5000, AN=0
PLN/PARAGRAPH
CLP/
```

A keyword specification of Y indicates that the SIF command should be created with a transformation matrix. The CIRCLE, ARC, SYMBOL, ELLIPSE, ELLIPTICAL ARC, TEXT, and PARAGRAPH commands have a form which contains a matrix. The following example contains SIF commands after SIF-out processing where the matrix form of the command is generated.

```
CIR/RA=500, CE=0, 0, MA=1. , 0. , 0. , 1.
ARC/CC, CE=1000, 1000, P1=1500, 1000, P2=1000, 1500,
  MA=1. , 0. , 0. , 1.
ELL/CE=2000, 2000, P1=2500, 2000, P2=2000, 2250,
  MA=1. , 0. , 0. , 1.
EAR/CE=3000, 3000, P1=3500, 3000, P2=3000, 3250,
  ST=90, SW=180, MA=1. , 0. , 0. , 1.
TLC/CD=4, JU=3
TXT/TH=100, TW=100, DR=4000, 4000, MA=1. , 0. , 0. , 1. , TEXT
TPC/CD=9, NU=1, JU=1, SP=50
PAR/FO=0, ID=0, TH=100, TW=100, DR=5000, 5000,
  MA=1. , 0. , 0. , 1.
PLN/PARAGRAPH
CLP/
```

## 2.30 PF - Define the IGDS Pattern File Name

The PF keyword allows you to define the IGDS pattern file specification to be used during SIF processing. The PF keyword is recognized by the ATG, GTA, TRI, and TRD processors. The recommended extension for the pattern file is CEL or DGN. The pattern file contains the definition for all patterns in the SIF file. The pattern file must exist and be a contiguous file. If patterns are defined with the SIF PATTERN command, the pattern file must be a cell library which contains all pattern cells used in SIF processing. If patterns are defined with the SIF LINE CHARACTERISTICS command, the pattern file must be a design file containing pattern type 5 elements. If the PF keyword is omitted, no pattern processing occurs. The format of the PF keyword follows:

PF=patfile

PF=patfile/AB

PF=patfile/FU

The patfile field defines the file specification for the IGDS pattern file. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

The recommended method of pattern processing is with the SIF PATTERN command. This method allows more options and is less cumbersome to use. This method requires the PF keyword to contain the file specification of an IGDS cell library which contains all pattern cell definitions used during SIF processing. This cell library can be the same file specified with the CL keyword or a different file. During SIF-in processing, a pattern type 5 element is added to the design file along with the element to be patterned. These elements are combined into a graphic group and assigned the next available graphic group number. After SIF-in processing, the pattern component elements can be generated with the IGDS PDV processor.

The other method of pattern processing is with the SIF LINE CHARACTERISTICS command. This method is supplied for compatibility with the initial releases of SIF and should not be used by the new SIF user. This method requires the PF keyword to contain the file specification of an IGDS design file which contains nothing but pattern type 5 elements after

sign bit in the coordinate fields of SIF binary commands is interpreted as magnitude. The read-outs of coordinates in the design file depend on the global origin and working units defined by the user.

A keyword specification of Y indicates that both positive and negative coordinates are allowed. The coordinates in the design file are assumed to range from -2147483648 to 2147483647. Any coordinates in the SIF ASCII file outside this range cause errors. The sign bit in the coordinate fields of SIF binary commands indicates whether the coordinate is positive or negative. The read-outs of coordinates in the design file depend on the global origin and working units defined by the user.

## 2.29 OV - Define the Overlay Processing Option

The OV keyword allows you to define the Overlay Processing option. The OV keyword is recognized by the GTA and TRO processors. This keyword allows you to process only elements on certain levels. If the OV keyword is not specified in the environment file, all levels are processed. The format of the OV keyword follows:

OV=ALL

OV=levels

A keyword specification of ALL indicates that elements on all levels should be processed. The levels field defines one or more levels to be processed. Individual level numbers must be separated by a comma. For example, the string "1,5,10" causes elements on levels 1, 5, and 10 to be processed. A range of levels can be selected by separating the starting and ending levels with a dash. For example, the string 5-10,20-30 causes elements on levels 5 through 10 and 20 through 30 to be processed. Any combination of individual or range selections can be specified. All level selections must be in the range of 1 to 63.

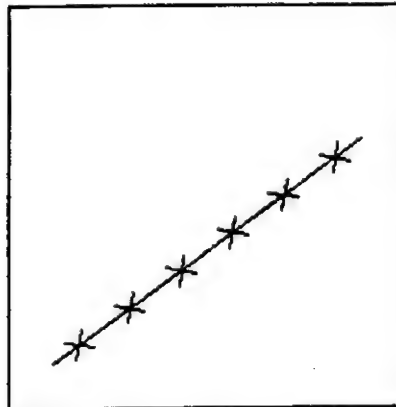


Figure 2-8. Example of Linear Patterning

If the PF keyword contains a pattern file specification and no switch, all primary and linear pattern elements are processed with the appropriate PATTERN command. If the PF keyword contains a cell library name, SIF PATTERN commands are created to define the pattern. If the PF keyword contains a design file name, SIF LINE CHARACTERISTICS commands are created to define the pattern. All pattern component elements are ignored. The SIF commands created when using this option follow:

```
CLA/5
PTN/PT=2, PS=0.25, PA=0., PD=0, 0, LINPAT
LST/OP, 1000, 1000, 8000, 6000
PTN/QF
```

The /AB switch indicates that the SIF file should contain the appropriate pattern definitions and the first two pattern components in the design file for that primary or linear pattern element. The two component elements can be used to determine the direction of the pattern and which side of the element was patterned. These elements are output as a SIF pattern set. The SIF commands created when using this option follow:

```
CLA/5
PTN/PT=2, PS=0.25, PA=0., PD=0, 0, LINPAT
LST/OP, 1000, 1000, 8000, 6000
CLA/1
LST/OP, 1000, 1000, 8000, 6000
LST/OP, 1528, 1083, 1639, 1750
PTN/QF
```

the design file header. These type 5 elements have a graphic group number which can range from 1 to 100. During SIF-in processing, the line or area pattern number in the LINE CHARACTERISTICS command is matched with a pattern type 5 element in the pattern file. This pattern type 5 element is added to the design file along with the element to be patterned. These elements are combined into a graphic group and assigned the next available graphic group number. After SIF-in processing, the pattern component elements may be generated with the IGDS PDV processor. This pattern file may contain only pattern type 5 elements which have a graphic group number from 1 to 100.

Several options are available to you for SIF-out processing. The graphics for the examples used in the following paragraphs is presented in Figure 2-8. If the PF keyword is not specified in the environment file, all primary, pattern component, and linear pattern elements are processed as they appear in the design file and are not output as a set. This option would be useful only to preserve the graphic picture. This option also processes graphic elements in the same order that they appear in the design file. The SIF commands created when using this option follow:

```
CLA/5
ASC/1
LST/OP, 1000, 1000, 8000, 6000
CLA/1
LST/OP, 1000, 1000, 8000, 6000
LST/OP, 1528, 1083, 1639, 1750
LST/OP, 1917, 1361, 1250, 1472
LST/OP, 2695, 1917, 2806, 2584
LST/OP, 3084, 2195, 2417, 2306
LST/OP, 3862, 2750, 3973, 3417
LST/OP, 4251, 3028, 3584, 3139
LST/OP, 5029, 3584, 5140, 4251
LST/OP, 5418, 3862, 4751, 3973
LST/OP, 6195, 4417, 6307, 5084
LST/OP, 6584, 4695, 5918, 4806
LST/OP, 7362, 5251, 7473, 5918
LST/OP, 7751, 5529, 7085, 5640
```

## 2.32 RA - Define the RANGE Command Option

The RA keyword allows you to specify whether a RANGE command should be created during SIF-out processing. The RA keyword is recognized by the GTA and TRO processors. The RANGE command is useful when translating graphics files to a system where the range of a graphic element is important and is not easily calculated from information in the SIF command. If the RA keyword is not specified in the environment file, no RANGE command is created. The format of the RA keyword follows:

RA=N

RA=Y

A keyword specification of N indicates that the range of each IGDS element processed should not be written to the SIF file.

A keyword specification of Y indicates that the range of each element processed should be written to the SIF file. The SIF RANGE command appears immediately before each SIF command which defines a graphic element. When complex strings or shapes are processed, the RANGE command is output immediately before the BEGIN STRING command, and no RANGE command is output for the component commands of the complex string. When a text node is processed, the RANGE command is output immediately before the PARAGRAPH command. An example of a SIF file with RANGE commands follows:

```
RNG/-500,-500,500,500
CIR/RA=500,CE=0,0
RNG/500,500,1500,1500
ARC/CC,CE=1000,1000,P1=1500,1000,P2=1000,1500
RNG/1500,1750,2500,2250
ELL/CE=2000,2000,P1=2500,2000,P2=2000,2250
TLC/CO=4,JU=3
RNG/4000,3950,4367,4150
TXT/TH=100,TW=100,OR=4000,4000,AN=0,TEXT
```

## 2.33 SD - Define the IGDS Seed File Name

The SD keyword allows you to define the IGDS seed file specification to be used during SIF processing. The SD keyword is recognized by the ATG and TRI processors. The recommended extension for the seed file is DGN. The seed file must exist and be a contiguous file. The format of the SD keyword follows:

SD=seedfile



The /FU switch indicates that the SIF file should contain the appropriate pattern definitions and all pattern components in the design file for that primary or linear pattern element. These elements are output as a SIF pattern set. The SIF commands created when using this option follow:

```
CLA/5
PTN/PT=2, PS=0.25, PA=0., PD=0,0, LINPAT
LST/OP, 1000, 1000, 8000, 6000
CLA/1
LST/OP, 1000, 1000, 8000, 6000
LST/OP, 1528, 1083, 1639, 1750
LST/OP, 1917, 1361, 1250, 1472
LST/OP, 2695, 1917, 2806, 2584
LST/OP, 3084, 2195, 2417, 2306
LST/OP, 3862, 2750, 3973, 3417
LST/OP, 4251, 3028, 3584, 3139
LST/OP, 5029, 3584, 5140, 4251
LST/OP, 5418, 3862, 4751, 3973
LST/OP, 6195, 4417, 6307, 5084
LST/OP, 6584, 4695, 5918, 4806
LST/OP, 7362, 5251, 7473, 5918
LST/OP, 7751, 5529, 7085, 5640
PTN/OF
```

### 2.31 PT - Define the Pad Text Option

The PT keyword allows you to specify the status of the Pad Text option. The PT keyword is recognized by the ATG and TRI processors. If the PT keyword is not specified in the environment file, all TEXT commands are blank padded to contain the number of characters specified in the TEXT CHARACTERISTICS command. The format of the PT keyword follows:

PT=N

PT=Y

A keyword specification of N indicates that the text in SIF TEXT commands should not be padded with blanks. All trailing blanks in the text string are ignored. The number of characters excluding trailing blanks is used as if that number had been specified in the TEXT CHARACTERISTICS command. If the entire text string is blank, a single blank character is output. If the range form of the text command is used, the actual number of characters in the text string excluding trailing blanks determines the text width.

A keyword specification of Y indicates that the text in SIF text commands should be padded with blanks so that it contains the number of characters specified in the TEXT CHARACTERISTICS command. This method is the only way to create a text string which contains trailing blanks. If the RANGE form of the TEXT command is used, the number of characters in the TEXT CHARACTERISTICS command determines the text width.



## 2.35 TC - Define the Track Changes Option

The TC keyword allows you to specify whether changes should be tracked during SIF processing. The TC keyword is recognized by the ATG, GTA, TRI, and TRO processors. This feature allows you to track changes made to the IGDS design file and DMRS database after SIF-in processing. If the TC keyword is not specified in the environment file, changes are not tracked. The format of the TC keyword follows:

TC=N

TC=Y

A keyword specification of N indicates that changes should not be tracked after SIF-in processing. The elements generated during SIF-in processing are defined as new elements. The SIF CHANGE command is not created during SIF-out.

A keyword specification of Y indicates that changes should be tracked after SIF-in processing. All graphic elements and database entities are defined as old elements. As each graphic element or entity is created, the new and modified bits of each element are cleared. After SIF-in processing, any modifications to the design file or database are reflected in the new and modified bits. All new elements added after SIF-in processing have the new bit set. Any elements changed have the modified bit set. During SIF-out processing, the SIF CHANGE command is output immediately before any element which has the new and/or modified bits set. The CHANGE command defines whether the change is for a database entity or a graphic element and whether the change is a new, modified, or new and modified element. For database entities, the CHANGE command appears immediately before the SIF IDENTIFIER command which defines the entity. For graphic elements, the CHANGE command appears immediately before the commands which define graphic elements. If both a CHANGE command and a RANGE command appear for the same element, the CHANGE command is output before the RANGE command. An example of a SIF file with CHANGE commands follows (SIF RANGE commands are also included to show the order of commands when both CHANGE and RANGE commands are requested.):

The seedfile field defines the file specification for the IGDS cell library. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

In addition to the header information required by all IGDS design files, the seed file can also contain any number of graphic elements. The entire seed file is copied into the output design file before SIF-in processing begins.

#### 2.34 SX - Define the SIF Binary File Name

The SX keyword allows you to define the SIF binary file specification to be used during SIF processing. The SX keyword is recognized by the APT, ASI, ASO, BTI, BTO, DLT, HLP, RCV, TRI, TRD, and TRU processors. The recommended extension for the SIF binary file name is BIN. The format of the SX keyword follows:

SX=filespec

The filespec field defines the file specification for the SIF binary file. The device, UIC, file name, extension, and version may be specified. Named directories and logical names may not be used. The device and UIC default to the user's device and UIC if not specified. The file name and extension must be specified. The version number defaults to the latest version if not specified.

The SIF binary file name must be specified in the environment file if you are processing in a single file mode or if an error results. When processing in a multiple file mode with a SIF index file, the SX keyword is ignored and can be omitted, since file specifications from the index file are used.

The SIF binary file is one of the input files to the ASO, BTO, HLP, and TRI processors and, therefore, must exist before these processors can be executed. The SIF binary file is one of the output files created by the ASI, BTI, and TRD processors. The DLT, RCV, and TRU processors only use the binary file specification to form other file specifications required during processing.

A keyword specification of ALL indicates that all files on the tape should be processed. The files field defines one or more tape files to be processed. Individual tape file numbers must be separated by a comma. For example, the string 1,5,10 would cause tape files 1, 5, and 10 to be processed. A range of tape files are selected by separating the starting and ending files with a dash. For example, the string 5-10,20-30 causes tape files 5 through 10 and 20 through 30 to be processed. Any combination of individual or range selections can be specified. All tape file selections must be in the range of 1 to 1600.

## 2.38 TN - Define the PARAGRAPH Command Format for SIF-out

The TN keyword allows you to specify which form of the PARAGRAPH command should be created during SIF-out processing. The TN keyword is recognized by the GTA and TRO processors. Two forms of the PARAGRAPH command are available. The ORIGIN form allows a paragraph to be defined by the origin, text size, and rotation angle. The RANGE form allows a paragraph to be defined by the top left point, bottom left point, and bottom right point where the three points define the range box for the paragraph. If the TN keyword is not specified in the environment file, the ORIGIN form of the PARAGRAPH command is created. The format of the TN keyword follows:

TN=ORIGIN

TN=RANGE

A keyword specification of ORIGIN indicates that SIF-out processing should create the PARAGRAPH command which defines the paragraph by the origin, text size, and rotation angle. This form of the PARAGRAPH command can be requested during both 2-D and 3-D processing. The format for the origin form of the PARAGRAPH command follows:

PAR/OR=xc,yc,TH=height,TW=width,AN=rota  
PAR/OR=xo,yo,zo,TH=height,TW=width,AN=rota

A keyword specification of RANGE indicates that SIF-out processing should create the PARAGRAPH command which defines the paragraph by the top left, bottom left, and bottom right vertices. This form of the PARAGRAPH command can be requested during both 2-D and 3-D processing. The format for the RANGE form of the PARAGRAPH command follows:

PAR/TL=tlx,tly,BL=blx,bly,BR=brx,bry  
PAR/TL=tly,tly,tlz,BL=blx,bly,blz,BR=brx,bry,brz

```

CHA/DB, MD
IDE/AS=1, CO=0, ID=1, KE=0
ASV/DE=?, ID=0, KE=0, ?1?CIRCLE?
RNG/-500, -500, 500, 500
CIR/RA=500, CE=0, 0
IDE/AS=1, CO=0, ID=1, KE=0
ASV/DE=?, ID=0, KE=0, ?1?ARC?
RNG/500, 500, 1500, 1500
ARC/CC, CE=1000, 1000, P1=1500, 1000, P2=1000, 1500
IDE/AS=1, CO=0, ID=1, KE=0
ASV/DE=?, ID=0, KE=0, ?1?ELLIPSE?
CHA/GR, MD
RNG/1500, 1750, 2500, 2250
ELL/CE=2000, 2000, P1=2500, 2000, P2=2000, 2250
CHA/DB, MD
IDE/AS=1, CO=0, ID=1, KE=0
ASV/DE=?, ID=0, KE=0, ?1?TEXT?
IDE/AS=2, CO=0, ID=1, KE=0
TLC/CO=4, JU=3
CHA/GR, MD
RNG/4000, 3950, 4367, 4150
TXT/TH=100, TW=100, OR=4000, 4000, AN=0, TEXT

```

## 2.36 TD - Define the Tape Density

The TD keyword allows you to specify the tape density of the SIF tape being processed. The TD keyword is recognized by the ATI, ATO, BTI, and BTO processors. If the TD keyword is not specified in the environment file, the tape density is assumed to be 1600 bpi. The format of the TD keyword follows:

```
TD=800
```

```
TD=1600
```

## 2.37 TF - Define the Tape Files to Process

The TF keyword allows you to define the tape files to be processed. The TF keyword is recognized by the ATI, ATO, BTI, and BTO processors. This keyword allows you to process specified files on the tape. If the TF keyword is not specified in the environment file, all files are processed. The format of the TF keyword follows:

```
TF=ALL
```

```
TF=files
```

A keyword specification of ALL indicates that all elements types should be processed. The types field defines one or more element types to be processed. Individual element types must be separated by a comma. For example, the string 3,4,6,11 would cause element types 3, 4, 6, and 11 to be processed. A range of element types can be selected by separating the starting and ending element types with a dash. For example, the string 3-7,11-17 would cause element types 3 through 7 and 11 through 17 to be processed. Any combination of individual or range selections can be specified.

## 2.39 TX - Define the TEXT Command Format for SIF-out

The TX keyword allows you to specify which form of the TEXT command should be created during SIF-out processing. The TX keyword is recognized by the GTA and TRD processors. Two forms of the TEXT command are available. The ORIGIN form allows a text string to be defined by the origin, text size, and rotation angle. The RANGE form allows a text string to be defined by the top left point, bottom left point, and bottom right point where the three points define the range box for the text. If the TX keyword is not specified in the environment file, the ORIGIN form of the TEXT command is created. The format of the TX keyword follows:

TX=ORIGIN

TX=RANGE

A keyword specification of ORIGIN indicates that SIF-out processing should create the TEXT command which defines the text by the origin, text size, and rotation angle. This form of the TEXT command can be requested during both 2-D and 3-D processing. The format for the ORIGIN form of the TEXT command follows:

TXT/OR=xc, yc, TH=height, TW=width, AN=rota  
TXT/OR=xo, yo, zo, TH=height, TW=width, AN=rota

A keyword specification of RANGE indicates that SIF-out processing should create the TEXT command which defines the text by the top left, bottom left, and bottom right vertices. This form of the TEXT command may be requested during both 2-D and 3-D processing. The format for the RANGE form of the TEXT command follows:

TXT/TL=tlx, tly, BL=blx, bly, BR=brx, bry  
TXT/TL=tlx, tly, tlz, O, BL=blx, bly, blz, BR=brx, bry, brz

## 2.40 TY - Define the Element Type Processing Option

The TY keyword allows you to define the Element Type Processing option. The TY keyword is recognized by the GTA and TRD processors. This keyword allows you to process only certain element types. If the TY keyword is not specified in the environment file, all types are processed. The format of the TY keyword follows:

TY=ALL

TY=types

### 3. SIF INDEX FILE

The SIF index file capability is provided for SIF processing in a multiple file environment. The index file contains the name of the SIF environment file, the name(s) of the data file(s) to be processed, and status codes for each file processed. The index file has the following file characteristics:

- o direct access
- o formatted
- o fixed length records (64 bytes/record)
- o listable.

The index file can be generated by any software task which will generate a file with the proper characteristics. There is also a SIF processor delivered with the SIF product which generates the index file and allows editing capabilities for existing index files. The NDX processor is described in the Standard Interchange Format Users Guide.

The first record in each index file must contain the name of the SIF environment file. All other records in the index file contain one of three types of data files for SIF processing: SIF ASCII files, SIF binary files, or IGDS design files. Data file types may not be mixed in the same index file. Each data file record also contains an input status code and output status code for the data file on that record. The input status indicates whether the data file was generated correctly. The output status indicates whether the data file could be translated by the next SIF processor.

There is also an internal status field used only by SIF processors which write status codes in the index file. The internal status field allows SIF processing to begin with the file which is being processed if the SIF processor is terminated during execution. The index file record format is presented in Figure 3-1.

Col. 1 - 40 File Name Specification	Col. 41 - 50 Input Status	Col. 51 - 60 Output Status	Col. 61 - 64 Internal
SIF environment file name	blank	blank	blank
Data file name	status	status	status
•	•	•	•
•	•	•	•
•	•	•	•
Data file name	status	status	status

Figure 3-1. Example of SIF Index File Format

## NOTES



#### 4. SIF ASCII COMMAND LANGUAGE

The SIF ASCII command language is composed of readable command descriptions which may be edited with any on-line edit process. The SIF ASCII command file has the following characteristics:

- o formatted
- o sequential
- o variable length records
- o listable.

SIF ASCII commands consist of a three-character keyword denoting the command type followed by a slash "/". These four characters must be the first four characters of each ASCII command. Following the slash, the ASCII command may require other descriptor data to define parameters for the command. The descriptor data consists of a two-character keyword, an equal "=", and one or more data values. The descriptor data must be separated by commas, and may appear in any order within the command. If ASCII data is required, such as a text string, it must always appear at the end of the command. The ASCII command may not contain embedded blanks except in ASCII text fields.

SIF ASCII command lines may contain 4 to 1024 characters. A command line of no more than 80 characters is recommended since it is easier to perform on-line edits. If an entire ASCII command does not fit in one record, a SIF ASCII continuation line can be used to separate the command into several records. A continuation line is denoted by blanks in the first four columns and the continued data beginning in column 5. Any number of continuation lines can be used. There are two restrictions for continuation lines. The first restriction is that the two-character descriptor data keyword and the "equal" or "comma" which follows must appear on the same record. The second restriction concerns ASCII data at the trailing text fields which must be continued to the next line. The trailing text fields must contain a nonblank character at the end of each ASCII record.

##### 4.1 Special SIF ASCII Commands

The SIF ASCII commands described in this section are provided to compute certain data for other SIF commands which are not easily calculated by the user. These special ASCII commands are used in conjunction with standard SIF commands to produce the desired results.

## NOTES

#### 4.1.2 GENERATE PARAGRAPH RANGE BLOCK Command (PA3/)

PA3/OR=xo, yo, AN=rot, TH=height, TW=width

The GENERATE PARAGRAPH RANGE BLOCK command may be used to generate the three vertices defining the range block on the GENERATE PARAGRAPH command. The GENERATE PARAGRAPH RANGE BLOCK command should appear immediately before the GENERATE PARAGRAPH command. If the GENERATE PARAGRAPH RANGE BLOCK command is used, the top left, bottom left, and bottom right vertices for the GENERATE PARAGRAPH command should not be specified. The GENERATE PARAGRAPH RANGE BLOCK command is not recognized in a 3-D SIF file.

The OR keyword indicates the origin of the paragraph in UORs and is required. The origin is used in conjunction with the active justification as defined by the PARAGRAPH CHARACTERISTICS command to determine the three vertices.

The AN keyword indicates the rotation angle in degrees measured counterclockwise from the x-axis and is optional. The rotation angle "rot" has a range of 0 to 360 and defaults to 0 if not specified. Negative rotation angles are not allowed. Rotation angles can be entered as decimal parts of a degree.

The TH and TW keywords indicate the text height and text width in UORs and are optional. The text height and text width default to 100 UORs if not specified.

#### 4.1.3 GENERATE TEXT LINE RANGE BLOCK Command (TX3/)

TX3/OR=xo, yo, AN=rot, TH=height, TW=width

The GENERATE TEXT LINE RANGE BLOCK command can be used to generate the three vertices defining the range block on the GENERATE TEXT LINE command. The GENERATE TEXT LINE RANGE BLOCK command should appear immediately before the GENERATE TEXT LINE command. If the GENERATE TEXT LINE RANGE BLOCK command is used, the top left, bottom left, and bottom right vertices for the GENERATE TEXT LINE command should not be used. The GENERATE TEXT LINE RANGE BLOCK command is not recognized in a 3-D SIF file.

The OR keyword indicates the origin of the text line in UORs and is required. The origin is used in conjunction with the active justification as defined by the TEXT LINE CHARACTERISTICS command to determine the three vertices.

The AN keyword indicates the rotation angle in degrees measured counterclockwise from the x-axis and is optional. The rotation angle "rot" has a range of 0 to 360 and defaults to 0 if not specified. Negative rotation angles are not allowed. Rotation angles can be entered as decimal parts of a degree.

The TH and TW keywords indicate the text height and text width in UORs and are optional. The text height and text width default to 100 UORs if not specified.

#### 4.1.1 MATRIX Command (MTX/)

MTX/MX, AN=rot, XS=xsc, YS=ysc  
MTX/MY, AN=rot, XS=xsc, YS=ysc

The MATRIX command can be used to generate the transformation matrix for the GENERATE SYMBOL command. The MATRIX command should appear immediately before the GENERATE SYMBOL command. If the MATRIX command is used, the matrix is not specified in the GENERATE SYMBOL command. The generated matrix remains the active matrix until another MATRIX command is entered. An MATRIX command with no keywords generates an identity matrix. If scaling, rotation, or mirroring is required, the GENERATE SYMBOL command requires a rotation matrix. The matrix is computed in order of scaling, rotating, and then mirroring as follows:

$$\begin{bmatrix} t11 & t12 \\ t21 & t22 \end{bmatrix}$$

t11 = my \* xsc \* COS (rot)  
t21 = my \* xsc \* SIN (rot)  
t12 = my \* ysc \* -SIN (rot)  
t22 = my \* ysc \* COS (rot)

where: mx - Indicator for mirroring about the x-axis  
(1-no mirror, -1-mirror).  
my - Indicator for mirroring about the y-axis  
(1-no mirror, -1-mirror).  
rot - Rotation angle.  
xsc - X scale factor.  
ysc - Y scale factor.

The MX or MY keyword indicates that the matrix is to include mirroring and is optional. MX indicates mirroring about the x-axis through the graphic elements origin. MY indicates mirroring about the y-axis through the graphic elements origin. If MX or MY is not present, no mirroring is included in the matrix.

The AN keyword indicates the rotation angle in degrees measured counterclockwise from the x-axis and is optional. The rotation angle "rot" has a range of 0 to 360 and defaults to 0 if not specified. Negative rotation angles are not allowed. Rotation angles may be entered as decimal parts of a degree.

The XS and YS keywords indicate the X scale factor and Y scale factor and are optional. The scale factor "xsc" or "ysc" has a range of 0 to 10 and defaults to 1 if not specified. Negative scale factors are not allowed. Scale factors can be entered as decimal values.

## 5. SIF BINARY COMMAND LANGUAGE

The SIF binary command language is composed of 32-bit binary words which require a special processor in order to edit the file. The SIF binary command file has the following characteristics:

- o unformatted
- o sequential
- o fixed length physical records (1024 bytes/record)

SIF binary commands consist of one 32-bit word containing header information for the command. The header word is described in Figure 5-1. Some binary commands also require additional information which follows the header. If ASCII data is required, such as a text string, it must always appear at the end of the command and must be blank padded to a 32-bit boundary.

type	#words
subtype	value

Figure 5-1. SIF Binary Header Word

SIF binary commands can contain 1 to 256 32-bit words. Each physical record in the binary file contains 256 32-bit words and can contain several binary commands. Binary commands can cross physical record boundaries, and the last physical record must be padded with a SIF PAD command. (See Section 5.1.) The PAD command can also be used to prevent binary commands from crossing physical record boundaries.

type - Command type (0-255)

#word - Number of 32-bit words in the command excluding the command header (0-225). If the command crosses physical record boundaries, the number of words includes those on the next record also.

## NOTES

## 5.2 SIF Floating Point Format

Certain SIF binary commands require floating point values. Due to the various floating point formats from system to system, it is necessary to define one format for SIF binary commands. You may decide to use the SIF ASCII command language to bypass the binary floating point format.

subtype - Command subtype (0-255). This field is used for commands with multiple options.

value - Command value (0-255). This field is used for commands where the value can be expressed in 8 bits.

### 5.1 SIF Coordinate Format

SIF coordinate data is stored into 32-bit integers. You have the option of declaring coordinate values to be signed or unsigned values. Signed and unsigned coordinate data may not be mixed within the same SIF file.

Signed coordinate values have a range of -2147483648 to 2147483647. Signed values indicate that the most significant bit of the 32-bit value determines whether the value is positive or negative. The signed value format is presented in Figure 5-2.

Unsigned coordinate values that have a range of 0 to 4294967295. Unsigned values indicate that the entire 32 bits of the value determine the magnitude of the value, and all values are positive.

SIF coordinates must be specified in units of resolution (UDRs). The entire design plane is assumed to be a cube with 4294967296 UDRs along each side. You can define working units (feet, inches, etc.) to be related to the UDRs through the SIF environment file. (See Section 2.9.) Working units allow you to manipulate the graphics file more easily after it is created.

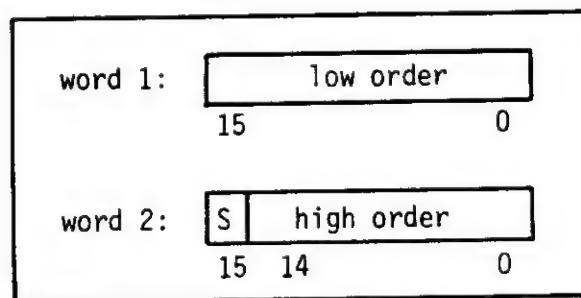


Figure 5-2. SIF Signed Value Format



## 6. SIF INTERFACE LIBRARIES

The SIF ASCII Command Library (ASCLIB) and the SIF Binary Command Library (BINLIB) consist of FORTRAN subroutines which allow the user to generate a file of SIF ASCII or binary commands. If ASCLIB is used, extra processing is required to convert the file of ASCII commands into a file of binary command, since the SIF tasks which read and write the graphic file and database recognize only SIF binary commands. The advantage of using an ASCII file is that it is in a readable format and is editable. The following subsection describes the utility subroutines included in the interface libraries. The subroutines which generate specific commands are presented in Sections 7 through 9 with the command format descriptions.

### 6.1 Image Linking

The ASCLIB and BINLIB interface libraries are interchangeable. The same FORTRAN task may generate either an ASCII or binary file of SIF commands depending upon which library is specified at link time. An example of the VAX LINK command file is presented in Figure 6-1.

TASK.LNK

```
$LINK/MAP TASK-  
$ WRK_DD_SIF:ASCLIB/LIB
```

Figure 6-1. Example of VAX LINK Command File

## NOTES

### 6.2.2 SIFEOF - End SIF File

The SIFEOF subroutine ensures that the last record of the SIF file gets written to the file. If a binary file is being written, a PAD command is used to force the last record to be 1024 bytes. SIFEOF is not required if writing an ASCII file but is included in the interface library to ensure interchangeability.

CALL SIFEOF

### 6.2.3 SIFCLO - Close SIF File

The SIFCLO subroutine closes the output SIF ASCII or binary file. SIFCLO should be called when all SIF commands have been generated.

CALL SIFCLO

## 6.3 File Characteristics

The file characteristic subroutines can be used to define the format of the SIF ASCII or binary file to be generated. All characteristics have a default value and need not be set if you desire the default value. If any changes are required, you must define the characteristics before any subroutines are called which generate SIF commands. The file characteristic subroutines are described in the following subsections.

### 6.3.1 SIFCOL - ASCII Characters Per Record

The SIFCOL subroutine can be used to define the maximum number of characters allowed in each record of the output ASCII file. The number of characters per record has a range of 20 - 1024 characters. The SIFCOL subroutine is also included in the binary interface library to ensure the interchangeability of the ASCII and binary libraries.

CALL SIFCOL (nc)

nc - Integer\*2 number of characters per record (Default is 72.)

### 6.3.2 SIFMOD - Define Graphic Mode

The SIFMOD subroutine can be used to indicate whether the output SIF file is to be a 2-D or 3-D SIF file.

## 6.2 File Maintenance

You must open the SIF ASCII or binary file. You can call subroutines within the interface libraries to open the file, write the end of file, and close the file. The interface libraries contain subroutines which may be called to open the file, write the end of file, and close the file.

If you do not use the open subroutine supplied in the interface library, the open statement must be changed if you change interface libraries. The file maintenance subroutines are described in the following subsections.

The SIF ASCII file is opened with the characteristics of formatted and sequential. The SIF binary file is opened with the characteristics of unformatted and sequential. The default logical unit for the SIF file is 4. The logical unit can be changed by including the following lines in the user program:

```
COMMON/SIFLUN/SIFLUN
```

```
SIFLUN = LUN
```

SIFLUN is the Integer\*2 logical unit for the SIF file.  
LUN is the Integer\*2 desired logical unit for the SIF file.

### 6.2.1 SIFOPE - Open SIF File

The SIFOPE subroutine opens the output SIF ASCII or binary file and should be called before any other SIF interface subroutine.

```
CALL SIFOPE (lun,ncfil,filnam,irc)
```

lun - Integer\*2 logical unit number for the output SIF file. This value is used to open the file and is used for all subsequent writes to the file.

ncfil - Integer\*2 number of characters in the file name.

filnam - Byte array containing the file name. This array must contain at least one extra byte which SIFOPE sets to 0 for the open statement.

irc - Integer\*2 return code.  
0 - normal return  
1 - error opening file

att - Byte array containing the ASCII attribute name or number. The array is assumed to be an attribute number if it contains all numeric characters.

ncv - Integer\*2 number of characters in attribute value.

val - Byte array containing the ASCII attribute value.

#### 6.4.2 SIFMAT - Compute Transformation Matrix

The SIFMAT subroutine can be used to define the rotation matrix for the next symbol to be placed with the CMD43 subroutine. CMD43 is described in Section 8.4.3. A rotation angle of 0, scale factors of 1, and no mirroring produces an identity matrix.

CALL SIFMAT (rot, scale, matrix, mir)

rot - Real\*4 rotation angle in degrees

scale - Real\*4 array which contains the scale factor  
Scale(1) - X scale factor  
Scale(2) - Y scale factor.

matrix - Real\*4 array which contains the computed matrix. You should reserve four locations for a 2-D file and nine locations for a 3-D file.

mir - Integer\*2 mirror indicator  
0 - no mirroring  
1 - mirror about the x-axis through symbol origin  
2 - mirror about the y-axis through symbol origin

#### 6.4.3 SIFPTS - Define Vertices

The SIFPTS subroutine can be used to define vertices for a line string, shape, or curve without being required to have a buffer large enough to contain the entire array of vertices. CMD40 or CMD45 must be called before SIFPTS. CMD40 is described in Section 8.1.3, and CMD45 is described in Section 8.6.3. After CMD40 or CMD45 is called, SIFPTS can be called any number of times to include vertices in the SIF command. SIFPTS must also be called to indicate that all vertices have been sent. No other SIF interface subroutines may be called until all vertices have been sent. SIFPTS automatically generates any required SIF CONTINUE commands.

CALL SIFMOD (gmode)

gmode - Integer\*2 graphic mode.  
2 - 2-D graphic file (default)  
3 - 3-D graphic file.

### 6.3.3 SIFNEG - Define Coordinate Format

The SIFNEG subroutine can be used to indicate the format of 32-bit coordinate values. If negative coordinates are allowed, the high order bit is interpreted as a sign bit, and coordinate values can range from -2147483648 to 2147483647. If negative coordinates are not allowed, the high order bit is interpreted as magnitude, and coordinate values can range from 0 to 4294967295.

CALL SIFNEG (ind)

ind - Integer\*2 coordinate format indicator.  
0 - Negative coordinates are allowed (default).  
1 - Negative coordinates are not allowed.

## 6.4 Utility Subroutines

You can use the utility subroutines in conjunction with specified SIF interface subroutines to aid in computing or formatting data required for the SIF commands. The following subsections describe the utility subroutines available in the SIF interface libraries.

### 6.4.1 SIFATT - Define Attribute Value

You can use the SIFATT subroutine to define a single attribute value. The interface subroutine CMD83 must be called before SIFATT. CMD83 is described in Section 10.4.3. After CMD83 is called, SIFATT is called once for each attribute to be defined and, finally, is called to indicate that all attribute data has been sent. No other interface subroutine can be called until SIFATT has been called to indicate that all values have been sent. SIFATT automatically generates any required SIF CONTINUE commands and inserts delimiter characters.

CALL SIFATT (nca, att, ncv, val)

nca - Integer\*2 number of characters in attribute name or number. After all attribute data has been sent, SIFATT must be called with "nca" equal to -1.

#### 6.4.5 SIFTXT - Define Text Line Boundary

The SIFTXT subroutine can be used to compute the three vertices required to define the boundary of a text line. SIFTXT must be called before CMD60A if you wish the interface library to compute the three vertices. CMD60A allows you to generate a SIF TEXT LINE command and is described in Section 9.1.3.

CALL SIFTXT (length,origin,rot,th,tw,just,points)

length - Integer\*2 number of characters in text line

origin - Integer\*4 array containing the text line origin in UORs

rot - Real\*4 rotation angle in degrees

th - Integer\*4 text height in UORs

tw - Integer\*4 text width in UORs

just - Integer\*2 text line justification as described in Section 7.6.3

point - Integer\*4 array containing the three computed vertices.

#### 6.5 Internal Subroutines

There are several subroutines in the SIF interface libraries which are not directly called by the user. The subroutine names are presented in Figure 6-2 to indicate additional entry points which may appear in the map at task link time.

TASK.ODL

	.ROOT	TASK-SIF-*(OV1,OV2)
SIF:	.FCTR	QW:[15,157]ASCLIB/LB
OV1:	.FCTR	SUB1-SIF
OV2:	.FCTR	SUB2-SIF
	.END	

Figure 6-2. Subroutines in the SIF Interface Libraries

CALL SIFPTS (nv,points)

nv - Integer\*2 number of vertices in the "points" array. After all vertices have been sent, SIFPTS must be called with "nv" equal to -1.

points- Integer\*4 array containing the coordinates in UORs. Two locations define one vertex in a 2-D file, and three locations define one vertex in a 3-D file.

#### 6.4.4 SIFTND - Define Paragraph Boundary

The SIFTND subroutine can be used to compute the three vertices required to define the boundary of a text node. SIFTND must be called before CMD61A if you want the interface library to compute the three vertices. CMD61A allows you to generate a SIF PARAGRAPH command and is described in Section 9.2.3.

CALL SIFTND (length,origin,rot,th,tw,just,points,space,  
numlin)

length - Integer\*2 number of characters in the longest paragraph line

origin - Integer\*2 array containing the text node origin in UORs

rot - Real\*4 rotation angle in degrees

th - Integer\*4 text height in UORs

tw - Integer\*4 text width in UORs

just - Integer\*2 text node justification as defined in Section 7.7.3

points - Integer\*4 array containing the three computed vertices

space - Integer\*4 line spacing between paragraph lines in UORs

numlin - Integer\*2 number of paragraph lines.



## 7. GRAPHIC CHARACTERISTICS COMMANDS

This section describes all SIF commands which define the characteristics for SIF GRAPHIC PLACEMENT commands. Once a characteristic is defined, it remains in effect for all GRAPHIC PLACEMENT commands which follow until a command defining a new characteristic value is specified. All CHARACTERISTICS commands have an initial value which is used if no command which defines a certain characteristic is specified. The characteristics which may be specified follow:

- o overlay - define the active IGDS level
- o active Z - define the active Z range coordinates for 2-D placement commands
- o classification - define the active IGDS element class
- o association - begin or end graphic grouping
- o font - define the active IGDS text font
- o pattern - begin or end patterning
- o line characteristics - define the active line style, weight, or color
- o text characteristics - define the text length and justification
- o paragraph characteristics - define the text node maximum line length, number of lines, justification, and line spacing
- o temporary origin - define a temporary origin for relative placement of graphic elements
- o identifier - define database or user linkage
- o multiple identifier - begin or end multiple linkages

### 7.1 OVERLAY Command

The SIF OVERLAY command can be used to define the active IGDS level for graphic commands which follow. All graphic elements are placed on the active level. The active level may change at any time by specifying another OVERLAY command. The OVERLAY command can be used in both 2-D and 3-D SIF files. The

## NOTES

#### 7.1.4 OVERLAY Command Restrictions

The restrictions for the OVERLAY command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The OVERLAY command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.1.5 OVERLAY Command Examples

This section presents examples of the SIF OVERLAY command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command. The following example places a circle, elliptical arc, and text elements on different levels. Lines 1 and 2 define a circle element on level 3. Lines 3 and 4 define an elliptical arc element on level 11. Lines 5 through 7 define a text element on level 18.

1. OVR/3
2. CIR/RA=2500, CE=5000, 4500
3. OVR/11
4. EAR/CE=5000, 4500, P1=8000, 4500, P2=5000, 6000, ST=45, SW=180
5. OVR/18
6. TLC/CO=4, JU=4
7. TXT/TH=1000, TW=1500, OR=5000, 7000, TEXT

The following example places a line string, arc, and ellipse elements. Since no OVERLAY command was specified, all elements are placed on level 1.

OVERLAY command may appear anywhere in the SIF file unless otherwise noted in the section covering OVERLAY command restrictions. The formats for the ASCII and the binary form of the OVERLAY command are presented in the following sections.

An IGDS design file recognizes levels in the range of 1 to 63. When creating an IGDS design file, any SIF overlay outside the range of 1 to 63 is changed to an IGDS level specified in the SIF environment file as the error level (EL keyword). Any graphic commands which appear before the first OVERLAY command are placed on level 1.

#### 7.1.1 OVERLAY Command - ASCII Form

The ASCII form of the OVERLAY command is indicated by a command type of OVR followed by a slash "/". The IGDS level number must appear after the slash. IGDS levels have a range of 1 to 63. In the following example, "level" defines the active IGDS level for graphic commands which follow:

OVR/level

#### 7.1.2 OVERLAY Command - Binary Form

The binary form of the OVERLAY command is indicated by a command type of 20 in the command header. The words to follow field must always be set to 0. The subtype field is not used. The value field contains the IGDS level number.

20	0
0	level

level - active IGDS level number (1-63).

#### 7.1.3 OVERLAY Command - Interface Form

The interface form allows you to generate an OVERLAY command by specifying only the IGDS level number. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF OVERLAY command.

CALL CMD20(level)

level - Integer\*2 active IGDS level number (1-63).

### 7.2.1 ACTIVE Z - ASCII Form

The ASCII form of the ACTIVE Z command is indicated by a command type of ACZ followed by a slash "/". Following the slash, the keywords and values to define the z low and z high range coordinates may be specified. The coordinates must be specified as unsigned integer values. If no characters follow the slash, the active z low is set to 0 and the active z high is set to 4294967295. The LD keyword indicates that the value for the z low range follows and is optional. If the LD keyword is omitted, the active z low is set to 0. In the following examples, "zlow" defines the active z low range coordinate for graphic commands which follow. The HI keyword indicates that the value for the z high range follows and is optional. If the HI keyword is omitted, the active z high is set to 4294967295. In the following examples, "zhigh" defines the active z high range coordinate for graphic commands which follow.

```
ACZ/LD=zlow,HI=zhigh
ACZ/
ACZ/LD=zlow
ACZ/HI=zhigh
```

### 7.2.2 ACTIVE Z Command - Binary Form

The binary form of the ACTIVE Z command is indicated by a command type of 21 in the command header. The words to follow field must always be set to 2. The subtype and value fields are not used. The z low range coordinate is contained in the second word, and the z high range coordinate is contained in the third word. The coordinates must be specified as unsigned integer values.

21	2
0	0
Z low	
Z high	

zlow - z low range coordinate (unsigned)

zhigh - z high range coordinate (unsigned)

1. LST/OP, 2000, 2000, 3500, 5000, 5000, 3000, 6500, 7000, 8000, 4000
2. ARC/CC, CE=5000, 3500, P1=7500, 3500, P2=2500, 3500
3. ELL/CE=5000, 4500, P1=8000, 4500, P2=5000, 6000

The following example attempts to place a paragraph on level 100. Since IGDS levels have a range of 1 to 63, the paragraph is placed on the error level specified with the EL keyword in the SIF environment file.

1. OVR/100
2. TPC/CO=12, JU=5, NU=5, SP=280
3. PAR/OR=5000, 4500, TH=600, TW=600
4. PLN/This is a
5. PLN/paragraph
6. PLN/which is
7. PLN/center
8. PLN/justified.
9. CLP/

## 7.2 ACTIVE Z Command

The SIF ACTIVE Z command can be used to define the active z low and z high range coordinates for graphic commands which follow. All graphic elements are placed with the active z range coordinates. The active z range coordinates can change at any time by specifying another ACTIVE Z command. The ACTIVE Z command may be used only in 2-D SIF files, since z range coordinates are meaningful in 3-D files. Active z range coordinates can range from 0 to 4294967295. The ACTIVE Z command may appear anywhere in the SIF file unless otherwise noted in the section covering ACTIVE Z command restrictions. The formats for the ASCII and the binary form of the ACTIVE Z command are presented in the following sections.

A 2-D IGDS design file has only x and y coordinates in the element format. However, both 2-D and 3-D IGDS elements have x, y, and z coordinates in the range area of each element header. The z low and z high range coordinates in 2-D elements may be used for any purpose by special applications (elevation, line weight, etc.). The z low range coordinate must be less than or equal to the z high range coordinate if the range is to be used for scanning purposes. Active z range coordinates must be specified in UORs. Any graphic commands which appear before the first ACTIVE Z command are placed with a z low of 0 and a z high of 4294967295.

used for description purposes and is not part of the command. The following example places a circle and closed line string with active z coordinates and an arc with default z coordinates. Line 1 defines the z low coordinate to be 100 and the z high coordinate to be 200. Lines 2 and 3 define the circle and closed line string to be placed. Line 4 resets the z low and z high to the default values. Line 5 defines the arc to be placed with the default z coordinates.

1. ACZ/LQ=100,HI=200
2. CIR/RA=100,CE=0,0
3. LST/SQ,-100,-100,-100,100,100,100,100,-100,-100,-100
4. ACZ/
5. ARC/P1=-200,0,P2=0,200,P3=200,0

The following example attempts to place a circle and closed line string with an active z low which is negative. Since active z coordinates must be positive, the z low coordinate is set to 0. The z high of 200 specified in the ACTIVE Z command is used as the z high coordinate.

1. ACZ/LQ=-100,HI=200
2. CIR/RA=100,CE=0,0
3. LST/SQ,-100,-100,-100,100,100,100,100,-100,-100,-100

### 7.3 CLASSIFICATION Command

The SIF CLASSIFICATION command can be used to define the active IGDS element class for graphic commands which follow. All graphic elements are placed with the active element class. The active element class can change at any time by specifying another CLASSIFICATION command. The CLASSIFICATION command may be used in both 2-D and 3-D SIF files. The classification command may appear anywhere in the SIF file unless otherwise noted in the section covering CLASSIFICATION command restrictions. The formats for the ASCII and the binary form of the CLASSIFICATION command are presented in the following sections.

An IGDS design file recognizes element classes in the range of 0 to 6. When creating an IGDS design file, any SIF classification outside the range of 0 to 6 is changed to an element class of 0. Any graphic commands which appear before first classification command are placed with an element class of 0. The IGDS element classes are presented in Table 7-1.

### 7.2.3 ACTIVE Z Command - Interface Form

The interface form allows you to generate an ACTIVE Z command by specifying only the z low and z high range coordinates. The subroutine formats the command and writes the command to the output SIF file. The coordinates must be specified as unsigned integer values. The following example describes the calling sequence required to generate a SIF ACTIVE Z command.

```
CALL CMD21(zlow,zhigh)
```

zlow - Integer\*4 active z low range coordinate  
(unsigned).

zhigh - Integer\*4 active z high range coordinate  
(unsigned).

### 7.2.4 ACTIVE Z Command Restrictions

The restrictions for the ACTIVE Z command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The ACTIVE Z command may appear anywhere in the SIF file except for the following conditions. The ACTIVE Z command is recognized only in 2-D SIF files.

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/).

### 7.2.5 ACTIVE Z Command Examples

This section presents examples of the SIF ACTIVE Z command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is



#### 7.3.4 CLASSIFICATION Command Restrictions

The restrictions for the CLASSIFICATION command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The classification command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.3.5 CLASSIFICATION Command Examples

This section presents examples of the SIF CLASSIFICATION command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command. The following example places a complex string and a text element using a combination of primary and construction class elements. The complex string has three component line strings which form a straight line. The first and third segments are placed as primary elements. The second segment is placed as a construction element. The text element is placed on top of the construction element. When the construction lines display is turned off, the second segment is not displayed but is still in the file for trace applications. The graphics for this example with the construction line display on and off is presented in Figure 7-1.

1. LAC/LT=2
2. CLA/O
3. BST/OP
4. LST/OP, 0, 0, 2000, 4000

Table 7-1. IGDS Element Classes

- 0 - primary element
- 1 - pattern component
- 2 - construction element
- 3 - dimensioning element
- 4 - primary rule element
- 5 - linear patterned element
- 6 - construction rule element

### 7.3.1 CLASSIFICATION Command - ASCII Form

The ASCII form of the CLASSIFICATION command is indicated by a command type of CLA followed by a slash "/". The IGDS element class must appear after the slash. IGDS element classes have a range of 0 to 6. In the following example, "class" defines the active element class for graphic commands which follow.

CLA/class

### 7.3.2 CLASSIFICATION Command - Binary Form

The binary form of the CLASSIFICATION command is indicated by a command type of 22 in the command header. The words to follow field must always be set to 0. The subtype field contains the IGDS element class. The value field is not used.

22	0
class	0

class - active IGDS element class (0-6).

### 7.3.3 CLASSIFICATION Command - Interface Form

The interface form allows you to generate a CLASSIFICATION command by specifying only the IGDS element class. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF CLASSIFICATION command.

CALL CMD22(class)

class - Integer\*2 active IGDS element class (0-6).

The following example attempts to place a line string and ellipse with an element class of 15. Since only element classes in the range of 0 to 6 are recognized, the elements are placed with an element class of 0.

1. CLA/15
2. LST/OP, 2000, 2000, 3500, 5000, 5000, 3000, 6500, 7000, 8000, 4000
3. ELL/CE=5000, 4500, P1=8000, 4500, P2=5000, 6000

#### 7.4 ASSOCIATION Command

The SIF ASSOCIATION command can be used to define the graphic group number for graphic commands which follow. Graphic groups allow simple graphic elements to be classified as a graphic set. These simple graphic elements can then be manipulated as if they were a single graphic element. All graphic elements are placed with the active graphic group number. The active graphic group number can change at any time by specifying another ASSOCIATION command. The ASSOCIATION command may be used in both 2-D and 3-D SIF files. The ASSOCIATION command may appear anywhere in the SIF file unless otherwise noted in the section covering ASSOCIATION command restrictions. The formats for the ASCII and the binary form of the ASSOCIATION command are presented in the following sections.

An IGDS design file recognizes graphic group numbers in the range of 0 to 65535. A graphic group number of 0 indicates that graphic commands which follow should not be placed in a graphic group. A graphic group number in the range of 1 to 65535 indicates that graphic commands which follow should be placed in a graphic group and defines the unique number to tie the graphic elements together. A graphic group number of -1 indicates that graphic commands which follow should be placed in a graphic group and the next available system-generated graphic group number should be used to tie the graphic elements together. When creating an IGDS design file, any graphic group number greater than 65535 is divided by 65535. The remainder, or "modulo", is the graphic group number. Any graphic commands which appear before the first ASSOCIATION command are placed with a graphic group of 0 which indicates that elements are not part of a graphic group.

##### 7.4.1 ASSOCIATION Command - ASCII Form

The ASCII form of the ASSOCIATION command is indicated by a command type of ASC followed by a slash "/". The graphic group number must appear after the slash. Graphic group numbers may range from -1 to 65535. In the following example, "gnum" defines the graphic group number for graphic commands which follow.

ASC/ggnum

5. CLA/2
6. LST/OP, 2000, 4000, 5000, 4000
7. CLA/0
8. LST/OP, 5000, 4000, 9000, 0
9. EST/
10. TLC/CQ=4, JU=5
11. TXT/OR=6000, 4000, TH=500, TW=500, TEXT

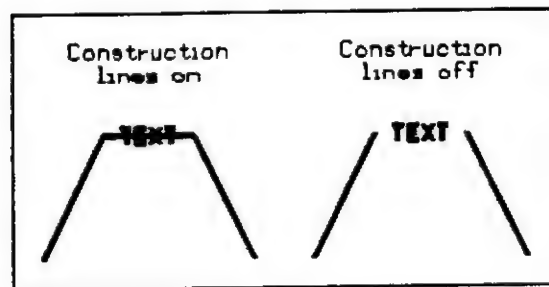


Figure 7-1. Example of Construction Line Display

- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.4.5 ASSOCIATION Command Examples

This section presents examples of the SIF ASSOCIATION command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command. The following example places a circle with some associated text elements where all elements are placed in a graphic group. Line 1 indicates that the next available graphic group number should be used. Lines 2 through 10 place the graphic elements. Line 11 turns the graphic grouping off.

1. ASC/-1
2. TLC/CO=20, JU=3
3. TXT/OR=500, 9000, TH=500, TW=500, 3. Circle by center
4. CIR/RA=2500, CE=5000, 4500
5. TLC/CO=1, JU=5
6. TXT/OR=5000, 4500, TH=500, TW=500, C
7. TLC/CO=1, JU=6
8. TXT/OR=6250, 4700, TH=500, TW=500, R
9. LAC/LS=1
10. LST/5500, 4500, 7500, 4500
11. ASC/0

The following example attempts to place several text strings with a graphic group number of -100. Since graphic group numbers have a range of -1 to 65535, the text elements are placed with no graphic grouping.

1. ASC/-100
2. TLC/CO=4, JU=4
3. TXT/TH=1000, TW=1500, OR=5000, 7000, TEXT
4. TLC/CO=3, JU=4
5. TXT/TH=750, TW=500, OR=5000, 5750, MAY
6. TLC/CO=4, JU=5
7. TXT/TH=500, TW=500, OR=5000, 4500, VARY
8. TLC/CO=2, JU=6
9. TXT/TH=750, TW=500, OR=5000, 3250, IN
10. TLC/CO=4, JU=6
11. TXT/TH=1000, TW=1500, OR=5000, 2000, SIZE
12. ASC/0

#### 7.4.2 ASSOCIATION Command - Binary Form

The binary form of the ASSOCIATION command is indicated by a command type of 23 in the command header. The words to follow field must always be set to 1. The subtype and value fields are not used. The graphic group number is contained in the second word.

23	1
0	0
GGNUM	

ggnum - graphic group number (-1 to 65535).

#### 7.4.3 ASSOCIATION Command - Interface Form

The interface form allows you to generate an ASSOCIATION command by specifying only the graphic group number. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF ASSOCIATION command.

```
CALL CMD23(ggnum)
```

ggnum - Integer\*4 graphic group number (-1 to 65535).

#### 7.4.4 ASSOCIATION Command Restrictions

The restrictions for the ASSOCIATION command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The ASSOCIATION command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)

CALL CMD24(font)

font - Integer\*2 active font number (0-126).

#### 7.5.4 FONT Command Restrictions

The restrictions for the FONT command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The FONT command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.5.5 FONT Command Examples

This section presents examples of the SIF FONT command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command. The following example places four text elements with four different text fonts. Since no font (FNT) command appears before the first TEXT command, it is placed with font 0. Lines 3 through 5 place a text element with font 1. Lines 6 through 8 place a text element with font 7. Lines 9 through 11 place a text element with font 41. The resulting graphics for this example are presented in Figure 7-2.

1. TLC/CO=16, JU=5
2. TXT/OR=9000,7000, TH=750, TW=750, Sample of font 0
3. FNT/1
4. TLC/CO=16, JU=5

## 7.5 FONT Command

The FONT command can be used to define the active IGDS font for graphic text commands which follow. All graphic text is placed with the active font. The active font can change at any time by specifying another FONT command. The FONT command may be used in both 2-D and 3-D SIF files. The FONT command may appear anywhere in the SIF file unless otherwise noted in the section covering FONT command restrictions. The formats for the ASCII and the binary form of the FONT command are presented in the following sections.

An IGDS design file recognizes text fonts in the range of 0 to 126. When creating an IGDS design file, any font outside the range of 0 to 127 is changed to 0. Any graphic text commands which appear before the first FONT command are placed with font 0.

### 7.5.1 FONT Command - ASCII Form

The ASCII form of the FONT command is indicated by a command type of FNT followed by a slash "/". The font number must appear after the slash. IGDS text fonts have a range of 0 to 255. In the following example, "font" defines the active text font for graphic text commands which follow:

FNT/font

### 7.5.2 FONT Command - Binary Form

The binary form of the FONT command is indicated by a command type of 24 in the command header. The words to follow field must always be set to 0. The subtype field is not used. The value field contains the text font number. In the following example, "font" defines the active text font for graphic text commands which follow:

24	0
0	FONT

font - active IGDS text font (0-126).

### 7.5.3 FONT Command - Interface Form

The interface form allows you to generate a FONT command by specifying only the text font number. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a FONT command:



## 7.6 PATTERN Command

The SIF PATTERN command can be used to define the active pattern for graphic commands which follow. When a pattern is specified, all graphic elements are placed with the active pattern until the pattern is turned off. The active pattern can change at any time by specifying another PATTERN command. The PATTERN command may be used in both 2-D and 3-D SIF files. SIF recognizes pattern types in the range of 0 to 5. The PATTERN command may appear anywhere in the SIF file unless otherwise noted in the section covering PATTERN command restrictions. The formats for the ASCII and the binary form of the PATTERN command are presented in the following sections.

SIF patterns are represented in IGDS design files as linear or area patterns. The types of patterns recognized in IGDS are presented in Table 7-2. SIF writes the element to be patterned and a pattern type 5 element which describes the pattern to the graphic file. These two elements are placed into a graphic group with the next available graphic group number. SIF does not generate the pattern component elements. As a postprocess to ATG or TRI, the IGDS Pattern Driver Task (PDV) must be executed if any elements were patterned in the SIF file.

Table 7-2. IGDS Pattern Types

- 0 - linear
- 1 - area
- 2 - scaled linear
- 3 - area (non-clip)
- 4 - segmented linear (multi)
- 5 - segmented linear (singles)

Linear patterns are applied from the first point in the element to the last point. As the pattern is applied, no spacing is added between the pattern components. Scaled linear patterns are identical to linear patterns except the scale factor is adjusted. The pattern begins and ends with a complete pattern cell. Multi-segmented patterning places as many complete pattern cells as possible on each segment of the element being patterned. Single-segmented patterning places only one complete pattern cell on each segment of the element being patterned.

5. TXT/OR=9000, 5000, TH=750, TW=750, Sample of font 1
6. FNT/7
7. TLC/CO=16, JU=5
8. TXT/OR=9000, 3500, TH=750, TW=750, Sample of font 7
9. FNT/41
10. TLC/CO=17, JU=5
11. TXT/OR=9000, 2000, TH=750, TW=750, Sample of font 23

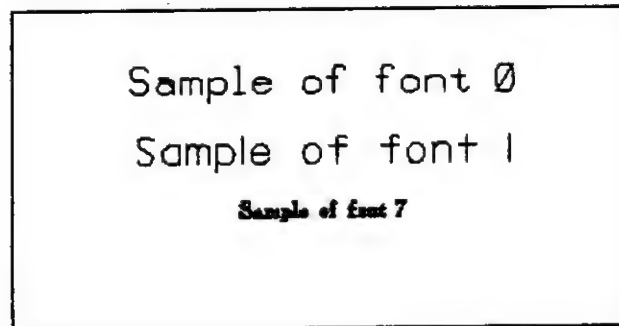


Figure 7-2. Example of Different Fonts

The following example attempts to place a text element with a text font of 300. Since IQDS fonts have a range of 0 to 126, the text element is placed with font 0.

1. FNT/300
2. TLC/CO=14, JU=3
3. TXT/OR=500, 9000, TH=500, TW=500, 19. Text fonts

25	*WORDS
PTYPE	0
SCALE *10000.	
ANGLE *10000.	
ROW SPACING	
COLUMN SPACING	
NAME	

25	*WORDS
255	0

type     -   pattern type (0-5).  
 scale    -   pattern scale. The pattern scale is multiplied  
           by 10000 and stored as an integer.  
 angle    -   pattern angle. The pattern angle is multiplied  
           by 10000 and stored as an integer.  
 row       -   row spacing in UORs.  
 column   -   column spacing in UORs.  
 name     -   pattern name.

### 7.6.3 PATTERN Command - Interface Form

The interface form allows you to generate a PATTERN command by specifying only the pattern number. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF PATTERN command.

```
CALL CMD25(type, scale, angle, delta, nc, name)
```

```
CALL CMD25A
```

type    -   Integer\*2 pattern type (0-5).

#### 7.6.1 PATTERN Command - ASCII Form

The ASCII form of the PATTERN command is indicated by a command type of PTN followed by a slash "/". Following the slash, the keywords and values to define the pattern type, angle, scale, delta, and name may be specified. The PT keyword indicates that the value for the pattern type follows and is optional. If the PT keyword is omitted, the pattern type is set to 0. In the following examples, "type" defines the pattern type. The PS keyword indicates that the value for the pattern scale follows and is optional. If the PS keyword is omitted, the pattern scale is set to 1. In the following examples, "scale" defines the pattern scale. The PA keyword indicates that the value for the pattern angle follows and is optional. If the PA keyword is omitted, the pattern angle is set to 0. In the following examples, "angle" defines the pattern angle. The PA keyword applies to area patterning only. The PD keyword indicates that the value for the pattern delta follows and is optional. The pattern delta consists of the row and column spacing in UDRs. If the PD keyword is omitted, the pattern delta is set to 0. In the following examples, "row,column" defines the row and column spacing. The PD keyword applies to area patterning only. The "name" field defines the name of the pattern and is required. This field must be the last field in the command. Only the first six characters of the pattern name are used. In the following examples, "name" defines the pattern name. The OF keyword indicates that the active pattern should be turned off. When this keyword is used, it should be the only keyword after the slash.

```
PTN/PT=type,PS=scale,PA=angle,PD=row,column,name
PTN/PT=type,PS=scale,name
PTN/PT=type,name
PTN/OF
```

#### 7.6.2 PATTERN Command - Binary Form

The binary form of the PATTERN command is indicated by a command type of 25 in the command header. The words to follow field must define the number of words following the command header. The subtype field contains the pattern type. The value field is not used. The pattern scale is contained in the second word. The pattern angle is contained in the third word. The pattern delta is contained in the fourth and fifth words. The pattern name begins in the sixth word.

patterning is required. After SIF-in processing is completed, the IGDS pattern driver (PDV) processor must be executed to create the pattern component elements. The resulting graphics with the pattern display both off and on is presented in Figure 7-3.

1. LAC/LT=1
2. PTN/PT=2, PS=.25, PA=0, PD=0, 0, LINPAT
3. LST/1000, 1000, 8000, 6000
4. PTN/OF
5. LAC/LT=0

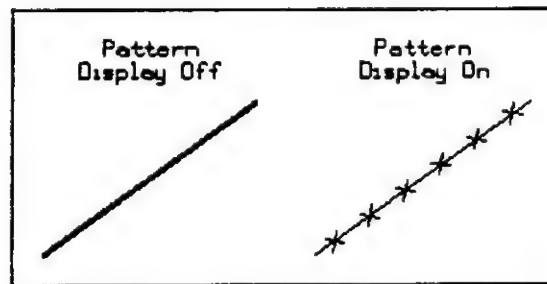


Figure 7-3. Example of SIF Linear Patterning

scale - Real\*4 pattern scale.

angle - Real\*4 pattern angle.

delta - Integer\*4 array of 2 words containing row spacing in the first word and the column spacing in the second word. The delta should be specified in UDRs.

nc - Integer\*2 number of characters in the pattern name.

name - Byte array containing the pattern name.

#### 7.6.4 PATTERN Command Restrictions

The restrictions for the PATTERN command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The PATTERN command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.6.5 PATTERN Command Examples

This section presents examples of the SIF PATTERN command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command.

The following example places a line element and indicates that the line should be patterned with the pattern cell named LINPAT. The pattern type of 2 indicates that scaled linear

## 7.7 LINE CHARACTERISTICS Command

The SIF LINE CHARACTERISTICS command can be used to define the active IGDS line style, weight, and color for graphic commands which follow. All graphic elements are placed with the active line characteristics. The active line characteristics can change at any time by specifying another LINE CHARACTERISTICS command. The LINE CHARACTERISTICS command may be used in both 2-D and 3-D SIF files. The LINE CHARACTERISTICS command may appear anywhere in the SIF file unless otherwise noted in the section covering LINE CHARACTERISTICS command restrictions. The formats for the ASCII and the binary forms of the LINE CHARACTERISTICS command are presented in the following sections.

An IGDS design file recognizes line styles in the range of 0 to 7. When creating an IGDS design file, any SIF line style outside the range of 0 to 7 is changed to an IGDS line style of 0. Any graphic commands which appear before the first line style specification are placed with a line style of 0. The IGDS line styles are presented in Table 7-3.

An IGDS design file recognizes line weights in the range of 0 to 31. When creating an IGDS design file, any SIF line weight outside the range of 0 to 31 is changed to an IGDS line weight of 0. Any graphic commands which appear before the first line weight specification are placed with a line weight of 0.

An IGDS design file recognizes line colors in the range of 0 to 255. When creating an IGDS design file, any SIF line color outside the range of 0 to 255 is changed to an IGDS line color of 0. Any graphic commands which appear before the first line color specification are placed with a line color of 0.

Table 7-3. IGDS Line Styles

- 0 - solid line
- 1 - dotted line
- 2 - medium dashed line
- 3 - long dashed line
- 4 - dash dot line
- 5 - short dashed line
- 6 - dash dot dot line
- 7 - long dash short dash line

The following example places a circle element and indicates the circle should be area patterned with the pattern cell named ARPAT. The pattern type of 1 indicates area patterning is required. After SIF-in processing is completed, the IGDS pattern driver (PDV) processor must be executed to create the pattern component elements. The resulting graphics with the pattern display both off and on is presented in Figure 7-4.

1. LAC/LT=2
2. PTN/PT=1, PS=1, PA=0, PD=0, 0, ARPAT
3. CIR/RA=2500, CE=4500, 3500
4. PTN/OF
5. LAC/LT=0

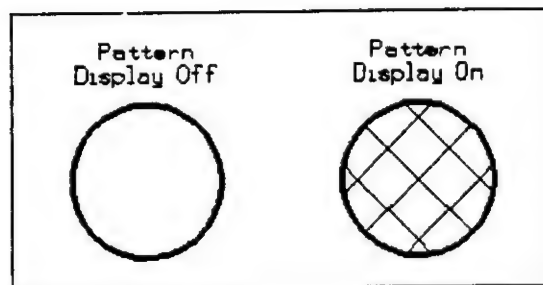


Figure 7-4. Example of SIF Area Patterning



### 7.7.2 LINE CHARACTERISTICS Command - Binary Form

The binary form of the LINE CHARACTERISTICS command is indicated by a command type of 26 in the command header. The words to follow field must be set to 0 if no pattern scale is specified and 1 if a pattern scale is specified. The subtype field contains the type of line characteristic being defined. The value field contains the line characteristics value. If a pattern scale is specified, it is contained in the second word. In the following example, "ctype" defines the line characteristic type, "cval" defines the line characteristic value, and "scale" defines the pattern scale for graphic commands which follow.

ctype - line characteristic type (0-4).

- 0 - line weight
- 1 - line style
- 2 - line color
- 3 - area pattern number
- 4 - linear pattern number

cval - line characteristic value (0-255).

scale - pattern scale. The pattern scale is multiplied by 10000 and stored as an integer.

26	0
CTYPE	CVAL

26	1
CTYPE	CVAL
SCALE *10000.	

### 7.7.3 LINE CHARACTERISTICS Command - Interface Form

The interface form allows you to generate a LINE CHARACTERISTICS command by specifying only the line characteristic type and value. The subroutine formats the command and writes the command to the output SIF file. The following examples describe the calling sequence required to generate a SIF LINE CHARACTERISTICS command.

In the initial releases of SIF, the LINE CHARACTERISTICS command also had the capability to define linear and area patterns for graphic elements. Although this capability has been preserved for initial SIF users, the SIF PATTERN command has the same capabilities and is much easier to use. The LINE CHARACTERISTICS form of patterning requires a pattern seed file in a format described in the section covering the SIF environment file's PF keyword. There is no method currently delivered to generate this pattern seed file. The Intergraph implementation of SIF recognizes pattern numbers in the range of 0 to 100. Any pattern number outside the range of 0 to 100 causes an error. A pattern number of 0 indicates the end of patterned elements.

#### 7.7.1 LINE CHARACTERISTICS Command - ASCII Form

The ASCII form of the LINE CHARACTERISTICS command is indicated by a command type of LAC followed by a slash "/". Following the slash, the keywords and values to define the line characteristics may be specified. One or more line characteristics may be specified in the same command. The LS keyword defines the line style and is optional. If the LS keyword is omitted, the line style is set to 0 which indicates a solid line. In the following examples, "style" defines the active line style for graphic commands which follow. The LT keyword defines the line weight and is optional. If the LT keyword is omitted, the line weight is set to 0 which indicates the smallest line weight. In the following examples, "weight" defines the active line weight for graphic commands which follow. The LC keyword defines the line color and is optional. If the LC keyword is omitted, the line color is set to 0. In the following examples, "color" defines the active line color for graphic commands which follow. The LP keyword defines the line pattern number and is optional. If the LP keyword is omitted, no linear patterning occurs. In the following examples, "lpat" defines the active linear pattern number for graphic commands which follow. The AP keyword defines the area pattern number and is optional. If the AP keyword is omitted, no area patterning occurs. In the following examples, "apat" defines the active area pattern number for graphic commands which follow. The PS keyword defines the pattern scale and is optional. If the PS keyword is omitted, the pattern scale is 1. In the following examples, "scale" defines the active pattern scale for graphic commands which follow.

```
LAC/LS=style
LAC/LT=weight
LAC/LC=color
LAC/LS=style,LT=weight,LC=color
LAC/LP=lpat,PS=scale
LAC/AP=apat
```

LAC/LT=0  
 LST/OP, 1000, 7000, 10000, 7000  
 LAC/LT=2  
 LST/OP, 1000, 6000, 10000, 6000  
 LAC/LT=4  
 LST/OP, 1000, 5000, 10000, 5000  
 LAC/LT=6  
 LST/OP, 1000, 4000, 10000, 4000  
 LAC/LT=8  
 LST/OP, 1000, 3000, 10000, 3000  
 LAC/LT=10  
 LST/OP, 1000, 2000, 10000, 2000

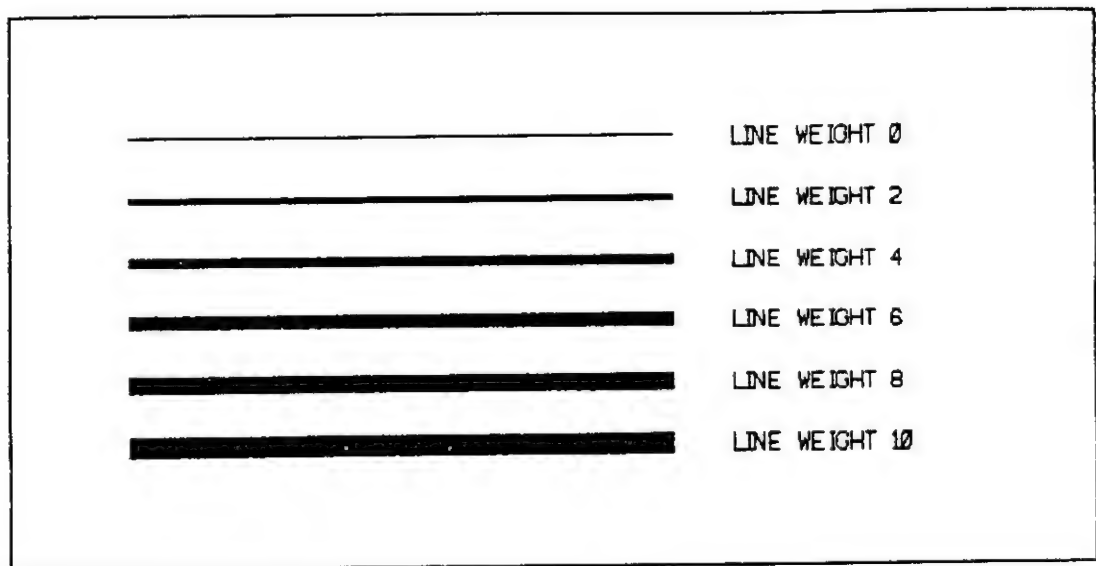


Figure 7-5. Example of Various Line Weights

CALL CMD26 (ctype,cval)

CALL CMD26P(ctype,cval,scale)

ctype - Integer\*2 line characteristic type.  
0 - line weight  
1 - line style  
2 - line color  
3 - area pattern number  
4 - linear pattern number

cval - Integer\*2 line characteristic value (0-255).

scale - Real\*4 pattern scale.

#### 7.7.4 LINE CHARACTERISTICS Command Restrictions

The restrictions for the LINE CHARACTERISTICS command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The LINE CHARACTERISTICS command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.7.5 LINE CHARACTERISTICS Command Examples

This section presents examples of the SIF LINE CHARACTERISTICS command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. The following example places several two-point line strings with various line weights. The resulting graphics is presented in Figure 7-5.

The following example attempts to place a line string with a linear pattern of 200. Since the pattern number of 200 is outside the range of 1 to 100, an error results.

LAC/LP=200, PS=2.  
LST/2000, 7000, 8000, 7000  
LAC/LP=0

The following example places several two-point line strings with various line styles. The resulting graphics is presented in Figure 7-6.

```
LAC/LT=2
LAC/LS=0
LST/OP, 1000, 8000, 10000, 8000
LAC/LS=1
LST/OP, 1000, 7000, 10000, 7000
LAC/LS=2
LST/OP, 1000, 6000, 10000, 6000
LAC/LS=3
LST/OP, 1000, 5000, 10000, 5000
LAC/LS=4
LST/OP, 1000, 4000, 10000, 4000
LAC/LS=5
LST/OP, 1000, 3000, 10000, 3000
LAC/LS=6
LST/OP, 1000, 2000, 10000, 2000
LAC/LS=7
LST/OP, 1000, 1000, 10000, 1000
```

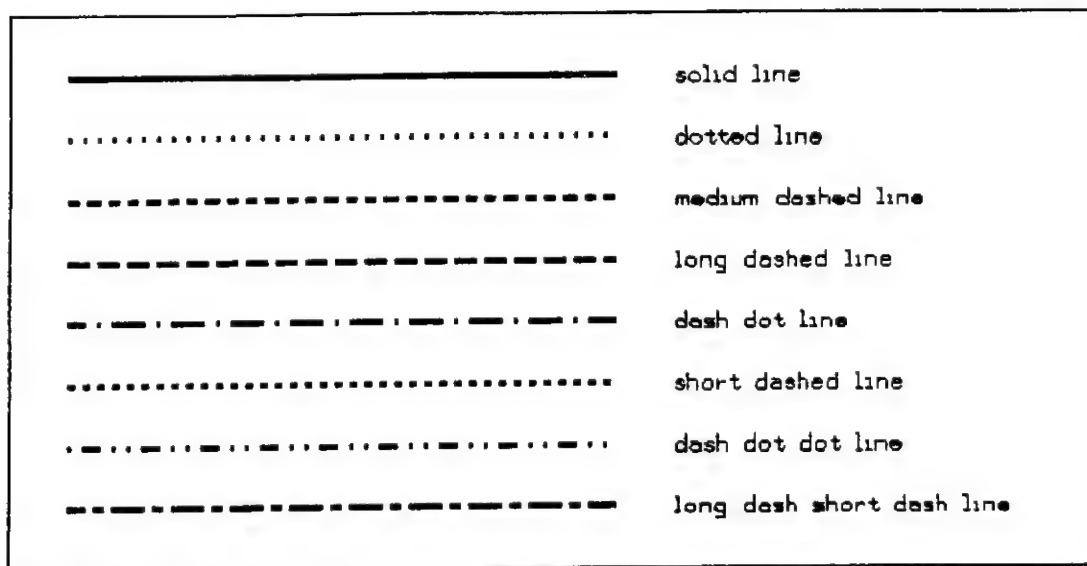


Figure 7-6. Example of Various Line Styles

characters are appended to the text in the TEXT command. Any graphic text commands which appear before the first TEXT CHARACTERISTICS command are placed with a length of 1 character. If PT=N, all text elements have a length equal to the number of text characters in the TEXT command. The number of characters in the previous TEXT CHARACTERISTICS command is ignored. If all characters are blank, the resulting text element has a single blank character.

### 7.8.1 TEXT CHARACTERISTICS - ASCII Form

The ASCII form of the TEXT CHARACTERISTICS command is indicated by a command type of TLC followed by a slash "/". Following the slash, the keywords and values to define the text length and justification for subsequent TEXT commands may be specified. The CO keyword defines the number of characters in subsequent TEXT commands and must be specified. IGDS text elements may contain from 1 to 255 characters. In the following examples, "count" defines the number of characters in graphic text commands which follow. The JU keyword defines the text justification and is optional. If the JU keyword is omitted, the justification is set to 3 which indicates bottom/left. In the following examples, "just" defines the justification for GRAPHIC TEXT commands which follow.

```
TLC/CO=count, JU=just
TLC/CO=count
```

### 7.8.2 TEXT CHARACTERISTICS Command - Binary Form

The binary form of the TEXT CHARACTERISTICS command is indicated by a command type of 27 in the command header. The words to follow field must always be set to 2. The subtype and value fields are not used. The number of characters in subsequent TEXT commands is contained in the second word. The justification is contained in the third word. In the following example, "count" defines the number of characters, and "just" defines the justification for graphic text commands which follow.

count - number of characters in subsequent text commands (0-255).

just - text justification (-14 to 9).

27	2
0	0
count	
just	

## 7.8 TEXT CHARACTERISTICS Command

The SIF TEXT CHARACTERISTICS command can be used to define the active text length and justification for graphic text commands which follow. All graphic text elements are placed with the active text length and justification. The active text length and justification can change at any time by specifying another TEXT CHARACTERISTICS command. The TEXT CHARACTERISTICS command may be used in both 2-D and 3-D SIF files. The TEXT CHARACTERISTICS command may appear anywhere in the SIF file unless otherwise noted in the section covering text characteristics command restrictions. The formats for the ASCII and the binary form of the TEXT CHARACTERISTICS command are presented in the following sections. IGDS text elements can be from 1 to 255 characters in length. Any TEXT commands which contain more than 255 characters are truncated to 255 characters. The text justification can be specified by using the SIF text justification code or the negative equivalent of the IGDS justification code. SIF justification codes and the equivalent IGDS justification codes are presented in Table 7-4. Any graphic text commands which appear before the first TEXT CHARACTERISTICS command are placed with a bottom/left justification.

Table 7-4. SIF Justification versus IGDS Justification

SIF Justification	IGDS Justification	Result
1	0	top/left
2	-1	center/left
3	-2	bottom/left
4	-6	top/center
5	-7	center/center
6	-8	bottom/center
7	-12	top/right
8	-13	center/right
9	-14	bottom/right

When SIF TEXT commands are translated into IGDS text elements, the length of the resulting text element depends upon the PT keyword in the SIF environment file. The PT keyword can have a value of Y or N. If PT=Y, all text elements have a length equal to the number of characters specified in the previous TEXT CHARACTERISTICS command. If necessary, blank



TLC/CD=5, JU=1  
 TXT/OR=-1000, 500, AN=0, JUST1  
 TLC/CD=5, JU=2  
 TXT/OR=-1000, 0, AN=0, JUST2  
 TLC/CD=5, JU=3  
 TXT/OR=-1000, -500, AN=0, JUST3  
 TLC/CD=5, JU=4  
 TXT/OR=0, 500, AN=0, JUST4  
 TLC/CD=5, JU=5  
 TXT/OR=0, 0, AN=0, JUST5  
 TLC/CD=5, JU=6  
 TXT/OR=0, -500, AN=0, JUST6  
 TLC/CD=5, JU=7  
 TXT/OR=1000, 500, AN=0, JUST7  
 TLC/CD=5, JU=8  
 TXT/OR=1000, 0, AN=0, JUST8  
 TLC/CD=5, JU=9  
 TXT/OR=1000, -500, AN=0, JUST9  
 LAC/LT=2  
 CIR/RA=10, CE=-1000, 500  
 CIR/RA=10, CE=-1000, 0  
 CIR/RA=10, CE=-1000, -500  
 CIR/RA=10, CE=0, 500  
 CIR/RA=10, CE=0, 0  
 CIR/RA=10, CE=0, -500  
 CIR/RA=10, CE=1000, 500  
 CIR/RA=10, CE=1000, 0  
 CIR/RA=10, CE=1000, -500

JUST1	JUST4	JUST7
JUST2	JUST5	JUST8
JUST3	JUST6	JUST9

Figure 7-7. Example of SIF Justifications

### 7.8.3 TEXT CHARACTERISTICS Command - Interface Form

The interface form allows you to generate a TEXT CHARACTERISTICS command by specifying only the number of characters and the justification. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF TEXT CHARACTERISTICS command.

```
CALL CMD27(count, just)
```

count - Integer\*2 number of characters in subsequent text commands (0-255).

just - Integer\*2 text justification (-14 to 9).

### 7.8.4 TEXT CHARACTERISTICS Command Restrictions

The restrictions for the TEXT CHARACTERISTICS command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The TEXT CHARACTERISTICS command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)

### 7.8.5 TEXT CHARACTERISTICS Command Examples

This section presents examples of the SIF TEXT CHARACTERISTICS command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. The following example places nine text elements using all possible text justifications. The SIF justification codes are used in the text commands. The circles are placed at the user-defined text origin to highlight the justifications. The resulting graphics for this example is presented in Figure 7-7.

contain only one line. In the following examples, "num" defines the number of lines in the text node. The JU keyword defines the text node justification and is optional. If the JU keyword is omitted, the justification is set to 1 which indicates top/left. In the following examples, "just" defines the justification for PARAGRAPH commands which follow. The SP keyword defines the line spacing between text node lines and is optional. If the SP keyword is omitted, the line spacing is 25 UORs. In the following examples, "space" defines the line spacing:

```
TPC/CO=count, NU=num, JU=just, SP=space
TPC/NU=num, JU=just, SP=space
TPC/JU=just
TPC/CO=count
```

#### 7.9.2 PARAGRAPH CHARACTERISTICS Command - Binary Form

The binary form of the PARAGRAPH CHARACTERISTICS command is indicated by a command type of 28 in the command header. The words to follow field must always be set to 4. The subtype and value fields are not used. The number of characters in the longest text node line is contained in the second word. The number of text node lines is contained in the third word. The justification is contained in the fourth word. The line spacing is contained in the fifth word. In the following example, "count" defines the number of characters in the longest text node line, "num" defines the number of text node lines, "just" defines the justification, and "space" defines the line spacing for PARAGRAPH commands which follow.

The following example attempts to place a text string at a justification of 10. Since only justifications in the range of -14 to 9 are recognized, the text string is placed with lower left justification.

```
TLC/CO=5, JU=10
TXT/OR=-1000, 1000, AN=0, JUST1
```

## 7.9 PARAGRAPH CHARACTERISTICS Command

The SIF PARAGRAPH CHARACTERISTICS command can be used to define the number of characters, number of lines, justification, and line spacing for PARAGRAPH commands which follow. All graphic paragraph elements are placed with the active paragraph characteristics. The paragraph characteristics can change at any time by specifying another PARAGRAPH CHARACTERISTICS command. The PARAGRAPH CHARACTERISTICS command may be used in both 2-D and 3-D SIF files. The PARAGRAPH CHARACTERISTICS command may appear anywhere in the SIF file unless otherwise noted in the section covering PARAGRAPH CHARACTERISTICS command restrictions. The formats for the ASCII and the binary forms of the PARAGRAPH CHARACTERISTICS command are presented in the following sections. IGDS text node lines may be from 1 to 255 characters in length. Any SIF PARAGRAPH commands which contain more than 255 characters are truncated to 255 characters. The text justifications may be specified using the SIF text justification code or the negative equivalent of the IGDS justification code. SIF justification codes and the equivalent IGDS justification codes are presented in Table 7-4. Any PARAGRAPH commands which appear before the first PARAGRAPH CHARACTERISTICS command are placed with a top/left justification.

### 7.9.1 PARAGRAPH CHARACTERISTICS - ASCII Form

The ASCII form of the PARAGRAPH CHARACTERISTICS command is indicated by a command type of TPC followed by a slash "/". Following the slash, the keywords and values to define the number of characters in the longest paragraph line, the number of lines, the justification, and the line spacing may be specified. The CO keyword defines the number of characters in the longest text node line and is optional. If the CO keyword is omitted, the longest text node line may contain up to 40 characters. IGDS text node lines may contain from 1 to 255 characters. In the following examples, "count" defines the number of characters in the longest text node line. The NU keyword defines the number of lines in the text node and is optional. If the NU keyword is omitted, the text node may

#### 7.9.4 PARAGRAPH CHARACTERISTICS Command Restrictions

The restrictions for the PARAGRAPH CHARACTERISTICS command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The PARAGRAPH CHARACTERISTICS command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)

#### 7.9.5 PARAGRAPH CHARACTERISTICS Command Examples

This section presents examples of the SIF PARAGRAPH CHARACTERISTICS command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. The following example places a text node with various text node characteristics. The first line indicates that each text node line may have up to 12 characters, the text node has 6 lines, the justification is center/center, and the line spacing is 50 UDRs. The resulting graphics are presented in Figure 7-8.

28	4
0	0
count	
num	
just	
space	

count - number of characters in longest text node line (0-255).

num - number of paragraph lines.

just - text justification (-14 to 9).

space - line spacing in UORs.

### 7.9.3 PARAGRAPH CHARACTERISTICS Command - Interface Form

The interface form allows you to generate a PARAGRAPH CHARACTERISTICS command by specifying only the number of characters in the longest text node line, the number of lines, the justification, and the line spacing. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF PARAGRAPH CHARACTERISTICS command.

CALL CMD28(count, num, just, space)

count - Integer\*2 number of characters in longest text node line (0-255).

num - Integer\*2 number of paragraph lines.

just - Integer\*2 justification (-14 to 9).

space - Integer\*4 line spacing in UORs.

The following example attempts to place a text node which contains six text node lines, but only three lines are specified in the PARAGRAPH CHARACTERISTICS command. An error message is printed to the message file, and an empty text node is placed.

```
TPC/CO=12, NU=3, JU=5, SP=50
PAR/OR=0, 0, TH=100, TW=100
PLN/This is a
PLN/sample of a
PLN/paragraph
PLN/which is
PLN/center
PLN/justified.
CLP/
```

## 7.10 TEMPORARY ORIGIN Command

The SIF TEMPORARY ORIGIN command can be used to define a temporary vertex in the graphic file as the temporary origin for graphic commands which follow. All graphic elements are placed relative to the active temporary origin. The active temporary origin coordinates can change at any time by specifying another TEMPORARY ORIGIN command. The TEMPORARY ORIGIN command may be used in both 2-D and 3-D SIF files. The TEMPORARY ORIGIN command may appear anywhere in the SIF file unless otherwise noted in the section covering TEMPORARY ORIGIN command restrictions. The formats for the ASCII and the binary form of the TEMPORARY ORIGIN command are presented in the following sections.

Temporary origin coordinates must be specified in UORs. Any graphic commands which appear before the first TEMPORARY ORIGIN command are placed with a temporary origin where all coordinates are 0.

### 7.10.1 TEMPORARY ORIGIN - ASCII Form

The ASCII form of the TEMPORARY ORIGIN command is indicated by a command type of TOR followed by a slash "/". Following the slash, the temporary origin coordinates may be specified. If no characters follow the slash, the temporary origin coordinates are set to 0. In the following examples, "tox, toy, toz" defines the temporary origin for graphic commands which follow.

```
TOR/tox, toy
TOR/tox, toy, toz
TOR/
```

TPC/CO=12, NU=6, JU=5, SP=50  
PAR/OR=0, 0, TH=100, TW=100  
PLN/This is a  
PLN/sample of a  
PLN/paragraph  
PLN/which is  
PLN/center  
PLN/justified.  
CLP/  
LAC/LT=2  
CIR/RA=10, CE=0, 0

A rectangular box with a thin black border. Inside the box, the text "This is a sample of a paragraph which is center justified." is displayed in a monospaced font, centered horizontally and vertically. The text is arranged in six lines: "This is a", "sample of a", "paragraph", "which is", "center", and "justified.".

This is a  
sample of a  
paragraph  
which is  
center  
justified.

Figure 7-8. Example of SIF Paragraph



#### 7.10.4 TEMPORARY ORIGIN Command Restrictions

The restrictions for the TEMPORARY ORIGIN command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The TEMPORARY ORIGIN command may appear anywhere in the SIF file except for the following conditions. The TEMPORARY ORIGIN command is recognized only in 2-D SIF files.

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.10.5 TEMPORARY ORIGIN Command Examples

This section presents examples of the SIF TEMPORARY ORIGIN command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. The following example places an ellipse and an elliptical arc using the same coordinates for the center point. By using a different temporary origin for each element, the elements are placed in different positions as shown in Figure 7-9.

```
LAC/LT=2
TOR/4500, 4500
ELL/CE=0, 0, P1=3000, 0, P2=0, 1500
TOR/13500, 4500
EAR/CE=0, 0, P1=3000, 0, P2=0, 1500, ST=45, SW=180
```

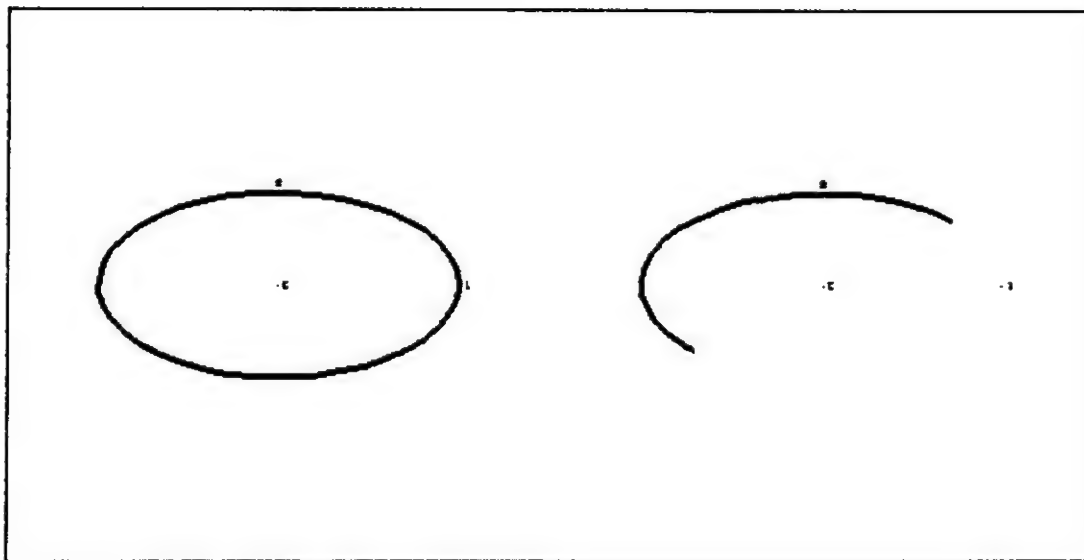


Figure 7-9. Example of SIF TEMPORARY ORIGIN Command

## 7.11 IDENTIFIER Command

The SIF IDENTIFIER command can be used to attach a linkage to a graphic element. The linkage may be a pointer to an entity in a DMRS database or simply be a graphic-only user linkage. The SIF IDENTIFIER command may also be used to insert a new entity into a DMRS database. The IDENTIFIER command may be used in both 2-D and 3-D SIF files. The IDENTIFIER command may appear anywhere in the SIF file unless otherwise noted in the section covering IDENTIFIER command restrictions. The formats for the ASCII and the binary form of the IDENTIFIER command are presented in the following sections.

Each linkage to the database requires four 16-bit words. Each user linkage is variable length but must be an even multiple of four 16-bit words. SIF pads ASCII user linkages with blanks and pads all other user linkages with binary zeroes. Individual user linkages cannot exceed 256 16-bit words. The different forms of linkages may be specified by the association type. Each association type requires different information as explained in the following paragraphs. An association type of 0 indicates that no linkage should be attached to graphic commands which follow. All other fields in the IDENTIFIER commands are ignored. All SIF placement commands appearing before the first command which attaches linkage cause elements to be placed with no linkage.

An association type of 1 indicates that a new entity must be inserted into the DMRS database. The number of the entity to be inserted must be specified. The linkage family class may range from 0 to 15. The occurrence number is ignored, and the next available occurrence number in the database is used. The next SIF command must be an ASSOCIATIVE VALUES command to define attribute values for the entity. The linkage to the database is attached to graphic elements which follow until another IDENTIFIER command is detected. The format for a database linkage is presented in Figure 7-10.

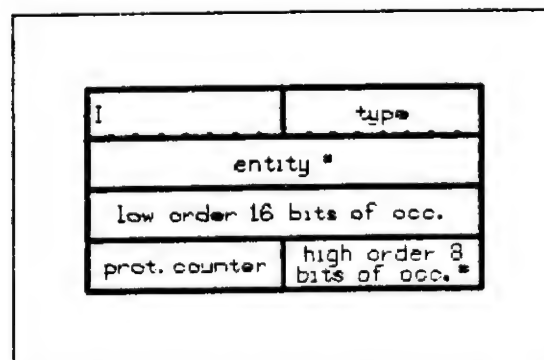


Figure 7-10. Database Linkage Format

An association type of 2 indicates that subsequent graphic elements should have a read/write linkage to an entity in the database. The number of the entity must be specified. The linkage family class may range from 0 to 15. The occurrence number may be specified or set to 0 to indicate that the previous linkage for the specified entity should be used. If specified, the occurrence number must include the protection counter.

An association type of 3 indicates that subsequent graphic elements should have a read only linkage to an entity in the database. The number of the entity must be specified. The linkage family class may range from 0 to 15. The occurrence number may be specified or set to 0 to indicate that the previous linkage for the specified entity should be used. If specified, the occurrence number must include the protection counter.

An association type of 4 indicates that subsequent graphic elements should have a user linkage. The user ID must be specified. The linkage family class may range from 0 to 15. The user information field must consist of ASCII characters. The format for a user linkage is presented in Figure 7-11.

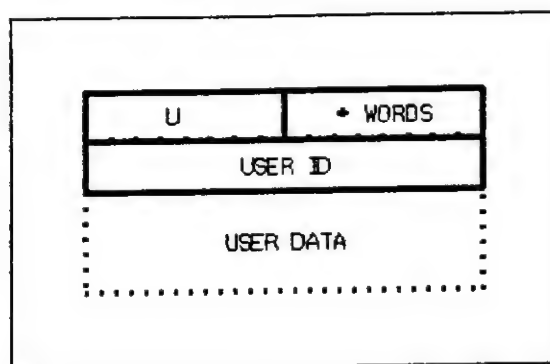


Figure 7-11. User Linkage Format

An association type of 5 indicates that subsequent graphic elements should have a user linkage. The user ID must be specified. The linkage family class may range from 0 to 15. The user information field must consist of 16-bit integers.

An association type of 6 indicates that subsequent graphic elements should have a user linkage. The user ID must be specified. The linkage family class may range from 0 to 15. The user information field must consist of 8-bit integers.

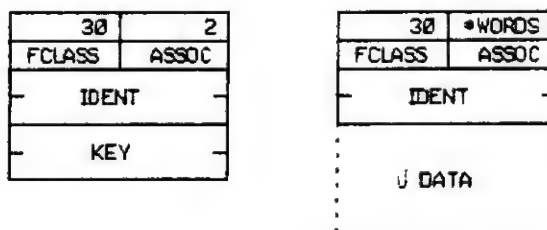
### 7.11.1 IDENTIFIER Command - ASCII Form

The ASCII form of the IDENTIFIER command is indicated by a command type of IDE followed by a slash "/". Following the slash, the keywords and values to define the association type, linkage family class, entity number, user ID number, occurrence number, and user information may be specified.

```
IDE/AS=0
IDE/AS=assoc,CO=fclass,ID=ident,KE=key
IDE/AS=assoc,CO=fclass,ID=ident,userdata
```

### 7.11.2 IDENTIFIER Command - Binary Form

The binary form of the IDENTIFIER command is indicated by a command type of 30 in the command header. The words to follow field must be set to 4 if the association type is less than 4. If the association type is greater than or equal to 4, the number of words to follow may vary. The subtype field contains the linkage family class. The value field contains the association type. In the following example, "fclass" defines the linkage family class, "assoc" defines the association type, "ident" defines the entity number or user ID, "key" defines the occurrence number, and "udata" defines the user linkage information.



fclass - linkage family class (0-15).  
 assoc - association type (0-6).  
 ident - entity number or user ID.  
 key - occurrence number.  
 data - user data.

### 7.11.3 IDENTIFIER Command - Interface Form

The interface form allows you to generate an IDENTIFIER command by specifying the critical information for the command. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF IDENTIFIER command. The CMD30 subroutine may be used only for database linkages. The CMD30A subroutine may be used only for user linkages. Either subroutine may be used to turn linkages off.

```
CALL CMD30 (fclass,assoc,ident,key)
```

```
CALL CMD30A(fclass,assoc,ident,nv,udata)
```

fclass - Integer\*2 linkage family class (0-15).

assoc - Integer\*2 association type (0-6).

ident - Integer\*2 identifier number.

key - Integer\*4 occurrence number.

nv - Integer\*2 number of values in user data.

udata - Array containing user data.

### 7.11.4 IDENTIFIER Command Restrictions

The restrictions for the IDENTIFIER command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The IDENTIFIER command may appear anywhere in the SIF file except for the following conditions:

- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)
- o immediately after a RANGE command (RNG/)

### 7.11.5 IDENTIFIER Command Examples

This section presents examples of the SIF IDENTIFIER command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is

used for description purposes and is not part of the command. For the examples of database linkages, assume that the database has entities 1, 2, and 3 where entity 1 is a parent of entity 2, and entity 2 is a parent of entity 3.

The following example inserts three entities into the database and places three line elements into the graphic file. Each line element points to a different entity in the database. Lines 1 and 2 insert an entity 1 and assign attribute 1 the value of ABC. Line 3 places the line element with the linkage to entity 1. Lines 4 and 5 insert an entity 2 and assign attribute 1 the value of DEF. Line 6 places the line element with the linkage to entity 2. Lines 7 and 8 insert an entity 3 and assign attribute 1 the value of GHI. Line 9 places the line element with the linkage to entity 3.

1. IDE/AS=1, CO=0, ID=1, KE=0
2. ASV/DE=?, ID=0, KE=0, ?1?ABC?
3. LST/OP, 0, 0, 100, 100
4. IDE/AS=1, CO=0, ID=2, KE=0
5. ASV/DE=?, ID=1, KE=0, ?1?DEF?
6. LST/OP, 100, 100, 200, 200
7. IDE/AS=1, CO=0, ID=3, KE=0
8. ASV/DE=?, ID=2, KE=0, ?1?GHI?
9. LST/OP, 200, 200, 300, 300

The following example places two line elements with a user linkage attached. Line 1 defines an ASCII user linkage to be placed on the line element generated by line 2. Line 3 defines a 16-bit user linkage to be placed on the line element generated by line 4.

1. IDE/AS=4, ID=63, THIS IS AN ASCII USER LINKAGE
2. LST/O, 0, 100, 100
3. IDE/AS=5, ID=100, 1, 2, 3
4. LST/100, 100, 200, 200

The following example attempts to use an association type of 10 to define an ASCII user linkage. Since association types range from 0 to 6, an error results.

1. IDE/AS=10, ID=63, THIS IS AN ASCII USER LINKAGE
2. LST/O, 0, 100, 100

## 7.12 MULTIPLE IDENTIFIER Command

The SIF MULTIPLE IDENTIFIER command can be used to attach more than one linkage to a graphic element. The MULTIPLE IDENTIFIER command is simply an on/off indicator to collect linkages to be attached to graphic elements. The linkages may

be pointers to entities in a DMRS database or simply graphic-only user linkages. All graphic elements are placed with the active linkage. The active linkage may change at any time by specifying another IDENTIFIER command or MULTIPLE IDENTIFIER command set. The MULTIPLE IDENTIFIER command may be used in both 2-D and 3-D SIF files. The MULTIPLE IDENTIFIER command may appear anywhere in the SIF file unless otherwise noted in the section covering MULTIPLE IDENTIFIER command restrictions. The formats for the ASCII and the binary forms of the MULTIPLE IDENTIFIER command are presented in the following sections.

The MULTIPLE IDENTIFIER commands must appear in pairs. The first MULTIPLE IDENTIFIER command must indicate the beginning of IDENTIFIER commands. The second MULTIPLE IDENTIFIER command must indicate the end of IDENTIFIER commands. The only SIF commands which may appear in a MULTIPLE IDENTIFIER set are the IDENTIFIER, ASSOCIATIVE VALUES, and CONTINUE commands. All database linkages contain four 16-bit words. User linkages may vary in length but must contain an even multiple of four 16-bit words. The length of the multiple linkage set in SIF is limited to 256 16-bit words. During SIF-in processing, the linkage set may contain any combination of database and user linkages. During SIF-out processing, only the first database linkage is processed due to the ordering procedure required to preserve the database structure. All user linkages are processed.

#### 7.12.1 MULTIPLE IDENTIFIER Command - ASCII Form

The ASCII form of the MULTIPLE IDENTIFIER command is indicated by a command type of MID followed by a slash "/". The keyword ON or OFF must appear after the slash to indicate the on/off status of the MULTIPLE IDENTIFIER command.

MID/ON  
MID/OFF

#### 7.12.2 MULTIPLE IDENTIFIER Command - Binary Form

The binary form of the MULTIPLE IDENTIFIER command is indicated by a command type of 31 in the command header. The words to follow field must always be set to 0. The subtype field indicates the status of the multiple identifier option. The value field is not used. In the following example, "stype" defines the on/off status of the multiple identifier option.



31	0
STYPE	0

stype - multiple linkage indicator  
 0 - off  
 1 - on

### 7.12.3 MULTIPLE IDENTIFIER Command - Interface Form

The interface form allows you to generate a MULTIPLE IDENTIFIER command by specifying only the on/off indicator. The subroutine formats the command and writes the command to the output SIF file. The following example describes the calling sequence required to generate a SIF MULTIPLE IDENTIFIER command:

```
CALL CMD31(stype)
```

stype - Integer\*2 multiple linkage indicator.  
 0 - off  
 1 - on

### 7.12.4 MULTIPLE IDENTIFIER Command Restrictions

The restrictions for the MULTIPLE IDENTIFIER command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The MULTIPLE IDENTIFIER command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before a CONTINUE command (CON/)

- o immediately after a CHANGE command (CHA/)
- o immediately after a RANGE command (RNG/)

#### 7.12.5 MULTIPLE IDENTIFIER Command Examples

This section presents examples of the SIF MULTIPLE IDENTIFIER command. The description of the example is presented first, followed by the example. Both valid and invalid examples are presented. The ASCII form of the command is used for all examples. Each command is preceded by a line number which is used for description purposes and is not part of the command.

The following example depicts the use of a MULTIPLE IDENTIFIER containing one database linkage and one user linkage. Lines 1 and 5 indicate the start and end commands for defining the multiple identifiers. Line 2 indicates that a database entity should be created and the database link added to subsequent graphic elements. Line 3 defines the value of attribute 1 to be ABC. Line 4 indicates that an ASCII user linkage should be added to subsequent graphic elements. Lines 6 and 7 indicate that a line string and circle should be placed in the graphic file. Both elements should contain the multiple linkage. Line 8 indicates that linkages to graphic elements should be turned off. Line 9 indicates that an arc should be placed in the graphic file with no linkage.

```

1.  MID/ON
2.  IDE/AS=1, CO=0, ID=1, KE=0
3.  ASV/DE=!, ID=0, KE=0, !1!ABC!
4.  IDE/AS=4, ID=63, USER LINKAGE
5.  MID/OF
6.  LST/0, 0, 100, 100
7.  CIR/RA=50, CE=0, 0
8.  IDE/AS=0, CO=0, ID=0, KE=0
9.  ARC/P1=0, -75, P2=0, 75, P3=75, 0

```

The following example places a line string with two database linkages attached. The first linkage points to a new entity 1, while the second linkage points to a new entity 2.

```

1.  MID/ON
2.  IDE/AS=1, CO=0, ID=1, KE=0
3.  ASV/DE=!, ID=0, KE=0, !1!ABC!
2.  IDE/AS=1, CO=0, ID=2, KE=0
3.  ASV/DE=!, ID=1, KE=0, !1!DEF!
5.  MID/OF
6.  LST/0, 0, 100, 100

```

The following example attempts to place a line string with two user linkages attached using an association type of 10. Since association types may range from 0 to 6, an error results.

1. MID/ON
2. IDE/AS=10, 1ST USER LINKAGE
3. IDE/AS=10, 2ND USER LINKAGE
4. MID/OF
5. LST/0, 0, 100, 100

## NOTES

## 8. GRAPHIC ELEMENT GENERATION COMMANDS

This section describes SIF graphic element generation commands. All SIF commands which generate graphics except graphic text are included. The description of each command presents the ASCII, binary, and interface form of the command as well as restrictions on usage in the SIF file and examples of the command. The graphic elements which can be generated are as follows:

o line string	IGDS LINE, LINE STRING, SHAPE
o circles	IGDS ELLIPSE
o circular arc	IGDS ARC
o symbol	IGDS CELL
o curve	IGDS CURVE
o ellipse	IGDS ELLIPSE
o elliptical arc	IGDS PARTIAL ELLIPSE
o conic	IGDS CONIC
o complex string	IGDS COMPLEX STRING OR SHAPE
o complex surface	IGDS SURFACE
o complex solid	IGDS CAPPED SURFACE
o point string	IGDS POINT STRING
o B-spline curve	IGDS B-SPLINE CURVE
o B-spline surface	IGDS B-SPLINE SURFACE
o circular truncated cone	IGDS CIRCULAR TRUNCATED CONE
o cylinder	IGDS CIRCULAR TRUNCATED CONE

### 8.1 LINE STRING Command - Generate Line, Line String, Shape

The SIF LINE STRING command can define a two-point line, a multi-point line, or a closed multi-point line (shape). The LINE STRING command can be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates must be specified for each vertex while a 3-D SIF file requires x, y, and z coordinates for each vertex. The SIF LINE STRING command can be continued with a SIF CONTINUE command or the SIF ASCII continuation line (four blanks at the beginning of the next

record). The LINE STRING command may appear anywhere in the SIF file unless otherwise noted in the section covering LINE STRING command restrictions. The formats for the ASCII and the binary forms of the LINE command are presented in the following sections.

The SIF LINE STRING command can generate three types of line strings: open line strings, a shape representing a solid, and a shape representing a hole. Open line strings with two vertices produce IGDS line elements. Open line strings with more than two vertices produce IGDS line string elements. IGDS line string elements are restricted to 101 vertices but SIF allows an unlimited number of vertices in the LINE STRING command. Any LINE STRING command having more than 101 vertices is broken into line strings of 101 vertices or less. Closed line strings produce IGDS shape elements. At least four (4) vertices are required to define a shape and the first and last vertex must be equal. SIF allows you to create closed line strings as either solids or holes.

#### 8.1.1 LINE STRING Command - ASCII Form

The ASCII form of the LINE STRING command is indicated by a command type of LST followed by a slash /. The slash can then be followed by one of three keywords and the coordinates of the vertices complete the ASCII LINE STRING command. The following are examples of a SIF ASCII LINE STRING command:

```
LST/OP, x1, y1, x2, y2, . . . , xn, yn
LST/HO, x1, y1, z1, x2, y2, z2, . . . , xn, yn, zn , x1, y1, z1
LST/SO, x1, y1, x2, y2, . . . , xn, yn, x1, y1
LST/x1, y1, , z1, x2, y2, z2
LST/x1, y1, x2, y2, . . . , xn, yn
```

The OP, SO, and HO keywords define the line string type and are optional. The OP keyword represents an open line string, the SO keyword represents a closed shape as a solid, and the HO keyword indicates a closed shape as a hole. If none of the keywords are specified, an open line string is assumed. The ASCII LINE STRING command can contain an unlimited number of vertices and can be continued with a SIF CONTINUE command or by the SIF ASCII continuation line (four blanks at the beginning of the next record). The vertices must be in the range of -2147483648 to 2147483647 (or 0-4B 4294967295 if no negative numbers are allowed) and must be in UORs.

#### 8.1.2 LINE STRING Command - Binary Form

The binary form of the LINE STRING command is indicated by a command type of 40 in the command header. The words to follow field must contain the number of Integer\*4 words to follow in the definition where each coordinate of a vertex is

one Integer\*4 word. The subtype field indicates the type of line string and the value field is always set to 0. In the following example, ltype defines the type of line string with "n" number of vertices.

4B	*WORDS
LTYPE	0
X1	
Y1	
X2	
Y2	
...	
XN	
YN	

ltype = 1 - open line string  
 2 - closed line string representing a solid  
 3 - closed line string representing a shape  
 4 - masked line string

### 8.1.3 LINE STRING Command - Interface Form

The interface form allows you to generate a LINE STRING command by specifying the line type, the number of vertices, and the coordinates of the vertices. The subroutine formats the command and writes the command to the output SIF file. The interface library automatically generates SIF CONTINUE commands where required or you can continue the LINE STRING command with a SIF CONTINUE command. The following example describes the calling sequence required to generate a SIF LINE STRING command.

```
CALL CMD40 (ltype,nv,points)
```

ltype - Integer\*2

1 - Open line string  
 2 - Closed line string representing a solid  
 3 - Closed line string representing a shape  
 4 - Construction element

nv - Integer\*2 number of vertices in the points array  
points - Integer\*4 array containing the coordinates of the line string in UORs.

The interface utility subroutine SIFPTS is provided to allow you to specify the coordinates for one line string with multiple calls. SIFPTS is useful if you do not know the number of vertices required for the line string or do not have enough buffer space to store all vertices for the line string. If SIFPTS is used, CMD40 must be called initially as follows:

```
CALL CMD40(1type,0,0)
```

#### B.1.4 LINE STRING Command Restrictions

The restrictions for the LINE STRING command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The LINE STRING command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o immediately before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)

#### B.1.5 LINE STRING Command Examples

This section presents examples of the SIF LINE STRING command. The description of the example is presented first followed by the example. Examples of both valid and invalid uses are given.

The following example places an open line string element, a closed line string as a solid and as a hole, and another open line string. The line strings are placed with different line styles and weights on varying levels. Figure 8-1 is a graphic depiction of the SIF ASCII commands.

```
LAC/LS=2  
LST/100,100,200,200  
LST/SD,150,200,300,600,900,500,150,200  
LAC/LS=0,LAC/LS=0,LT=1  
LST/HO,-1200,1100,-600,1700,-100,2200,400,2700,-1200,1100  
LAC/LT=0  
OVR/7  
LST/GP,1500,600,1000,-400,1500,-1000
```



The following example places a set of 3-D line strings on varying levels. Figure B-2 is a graphic depiction of the SIF ASCII commands.

```
OVR/10
LST/100,100,0,200,200,0,300,300,0,400,400,0
OVR/21
LST/SD,150,200,100,300,600,100,900,500,100,150,200,100
OVR/57
LST/H0,-1200,1110,-500,-600,1700,-500,-100,2200,-500,
    400,2700,-500,-1200,1100,-500
```

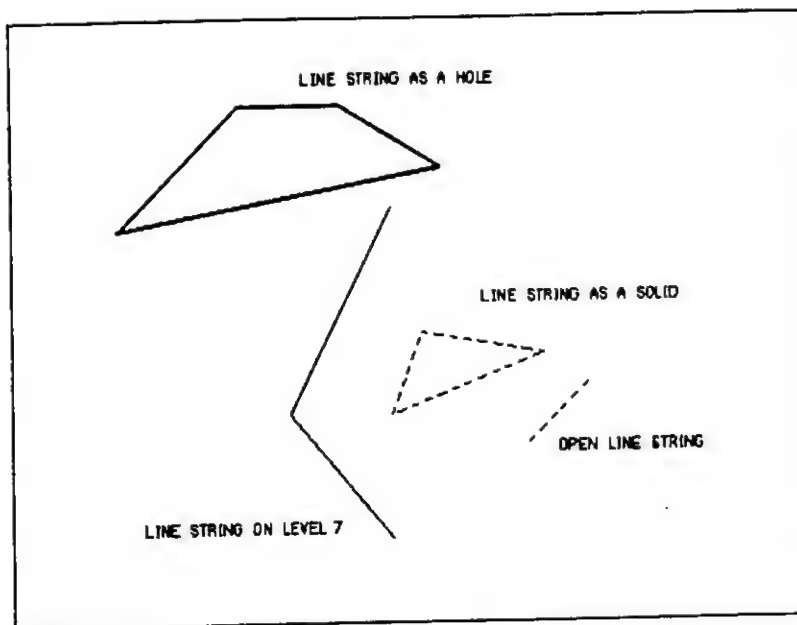


Figure 8-1. Example of SIF 2-D Lines and Line Strings

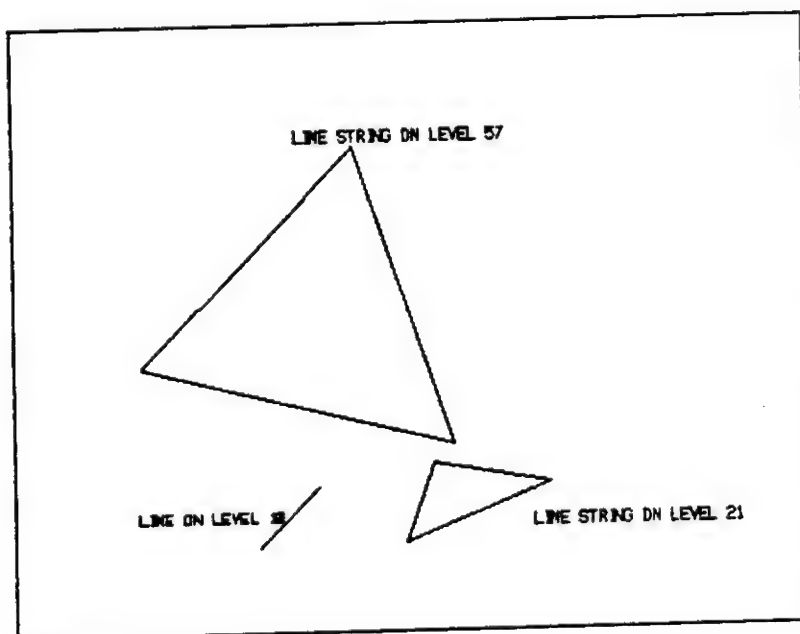


Figure 8-2. Example of SIF 3-D Lines and Line Strings

The following example attempts to place a line string inside the definition of a text node. Since the LINE STRING command is not allowed in this situation, it is flagged as an error and neither the line string or text node is placed in the output SIF file.

```
TPC/CO=10, NU=2, JU=3, SP=25
PAR/OR=100, 100, AN=45
PLN/NOW IS THE
LST/OP, 100, 100, 200, 200
PLN/FOR ALL
CLP/
```

## 8.2 CIRCLE Command - Generate Circle

The SIF CIRCLE command can define an IGDS ellipse element. The CIRCLE command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates must be specified when a vertex is used in the definition. A 3-D SIF file must contain an x, y, and z coordinate. There are two forms of the SIF CIRCLE command available. SIF files which are 2-D may use either the radius and center vertex definition or the center vertex - two vertices on the circle definition. Since it takes three vertices to define a plane, the valid definition for a circle in a 3-D SIF file is the center vertex-two vertices on circle form. Each of the two forms of the SIF CIRCLE command can also have a transformation matrix which is useful in engineering applications. The transformation matrix is recognized in both 2-D and 3-D SIF files where a 2-D file requires four terms and a 3-D file requires nine terms.

NOTE: The transformation matrix for the SIF CIRCLE command reflects pure rotation and should not be used as a scaling matrix.

The transformation matrix should always be input as a row major matrix as follows:

$$\begin{bmatrix} t(11) & t(12) & t(13) \\ t(21) & t(22) & t(23) \\ t(31) & t(32) & t(33) \end{bmatrix}$$

The SIF CIRCLE command generates an IGDS ellipse element and the vertices range from -2147483648 to 2147483647; however, care should be taken in the radius-center definition form that the radius length does not cause any point of the element to fall outside the IGDS design plane. All vertex coordinates should be in UORs (units of resolution). A CIRCLE command with radius set to 0 is a valid definition in IGDS and therefore acceptable in a SIF file. Since a SIF circle is supported in IGDS as a closed ellipse element, it can be patterned using the SIF PATTERNING command.

### 8.2.1 CIRCLE Command - ASCII Form

The ASCII form of the CIRCLE command is indicated by a command type of CIR followed by a slash (/). The slash can then be followed by one of the two different SIF ASCII forms that follow:

a) CIR/HO, RA=radius, CE=xc, yc, MA=t11, t21, t12, t22

b) CIR/CE=xc, yc, zc, P1=x1, y1, z1, P2=x2, y2, z2

In the SIF CIRCLE command form a, the RA keyword defines the length of the radius in UORs (units of resolution) while the CE keyword defines the coordinates of the center of the circle. The HO keyword defines the element as a hole. If this keyword is not specified, the element is defined as solid. The MA keyword defines the terms of the transformation matrix. The RA and CE keywords are required while the MA and HO keywords are optional.

In the SIF CIRCLE command form b, the CE keyword defines the coordinates of the center of the circle while the P1 and P2 keywords define arbitrary vertices which lie on the circumference of the circle. For 3-D SIF files, the coordinates of these two vertices may not be colinear. The MA keyword defines the terms of the transformation matrix. In this circle form, the CE, P1, and P2 keywords are required while the MA keyword is optional. If a transformation matrix is specified, it should reflect pure rotation. The matrix should always be input in row major form with terms in the range of -1. to 1.

### 8.2.2 CIRCLE Command - Binary Form

The binary form of the CIRCLE command is indicated by a command type of 41 in the command header. The words to follow field must contain the number of Integer\*4 words which are used to complete the definition. The subtype field indicates which of the four forms of the SIF CIRCLE command is being used while the value field is either 0 to indicate a solid or 1 to indicate a hole. Figure 8-3 illustrates four forms of the binary form of the CIRCLE command.

41	*WORDS
STYPE	STATUS
RADLG	
XC	
YC	

41	*WORDS
STYPE	STATUS
RADLG	
XC	
YC	
T11	
T21	
T12	
T22	

41	*WORDS
STYPE	STATUS
XC	
YC	
X1	
Y1	
X2	
Y2	

41	*WORDS
011	0
XC	
YC	
X1	
Y1	
X2	
Y2	
T11	•0000.
T21	•0000.
T12	•0000.
T22	•0000.

Figure 8-3. Four Forms of the CIRCLE Command - Binary Form

```

stype = 0 - radius-center form
        1 - center-two vertices form
       100 - radius-center form with transformation matrix
       101 - center-two vertices form with transformation
            matrix

```

```

status= 0 - solid
        1 - hole

```

NOTE: The values of the transformation matrix should be scaled by a factor of 10000 in order to preserve decimal accuracy.

### B.2.3 CIRCLE Command - Interface Form

The interface form allows you to generate a CIRCLE command by specifying the radius, center, and a flag indicating the presence of a matrix; or the second form can be generated by specifying the center, two vertices on the circumference, and a flag indicating the presence of a transformation matrix. The subroutine formats the command and writes it to a SIF output file. The following examples describe the calling sequence required to generate a SIF CIRCLE command:

```

CALL CMD41A(radius, center, value, mflag, matrix)

CALL CMD41B(center, point1, point2, value, mflag, matrix)

radius  - Integer*4 radius of circle in UORs

center  - Integer*4 array containing coordinates of
          circle center in UORs

point1  - Integer*4 array containing coordinates of point
          1 on circle in UORs

point2  - Integer*4 array containing coordinates of point
          2 on circle in UORs

value   - Integer*2 value indicating a solid or hole
          (0-solid; 1-hole)

mflag   - Integer*2 flag indicating presence of matrix
          (0-no matrix; 1-matrix)

matrix  - Real*4 array containing the scaled terms of the
          transformation matrix

```

#### 8.2.4 CIRCLE Command Restrictions

The restrictions for the CIRCLE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The CIRCLE command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o before a CONTINUE command (CON/)

#### 8.2.5 CIRCLE Command Examples

This section presents examples of the SIF CIRCLE command. The description of the example is presented first followed by the example. Both valid and invalid examples are presented and the ASCII form of the command is used for all examples.

The following example places four concentric circles on level 10 with different line display styles. Figure 8-4 is a graphic representation of the SIF ASCII commands.

```
OVR/10
LAC/LS=1
CIR/RA=10, CE=100, 100
LAC/LS=4
CIR/CE=100, 100, P1=120, 100, P2=100, 120, MA=1. , 0. , 0. , 1.
LAC/LS=3
CIR/RA=30, CE=100, 100, MA=1. , 0. , 0. , 1.
LAC/LS=2
CIR/CE=100, 100, P1=140, 100, P2=100, 60
```

The following example defines two 3-D circles, one as a solid with a matrix and one as a hole without a matrix. Figure 8-5 is a graphic representation of the SIF ASCII commands.

```
CIR/HO, CE=100, 0, 100, P1=200, 0, 100, P2=100, 0, 200
CIR/CE=0, 100, 200, P1=0, 200, 200,
P2=0, 100, 100, MA=. 707, 0. , 0. , 0. , 707, 0. , 0. , 0. , 1.
```

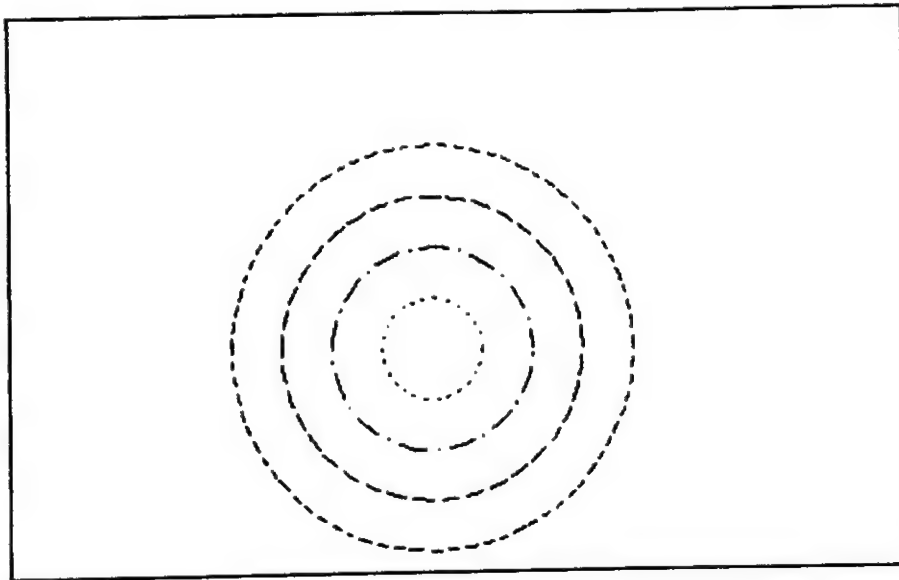


Figure 8-4. Example of SIF Concentric Circles

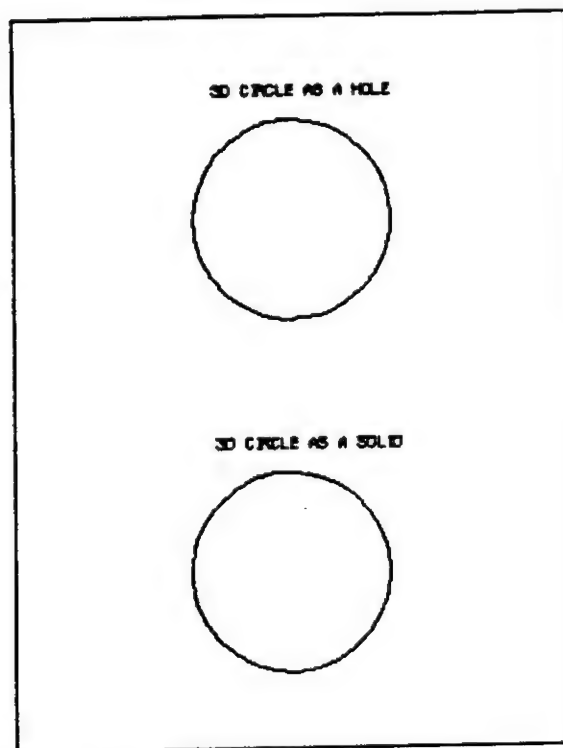


Figure 8-5. Examples of SIF 3-D Circles



The following example attempts to place a circle centered at point 2147483627,100 with a radius of 50000. Since the element defined at this origin is partially outside the design plane, its definition causes an error in placement. The second CIRCLE command is a proper definition of a circle which is to be patterned using a SIF PATTERN command. Figure 8-6 is a graphic representation of the patterned circle generated from the SIF ASCII commands.

```
CIR/RA=50000,CE=2147483627,100
PTN/PT=1,PS=1.,CIRPAT
CIR/CE=100,100,P1=100,200,P2=0,100
PTN/OF
```

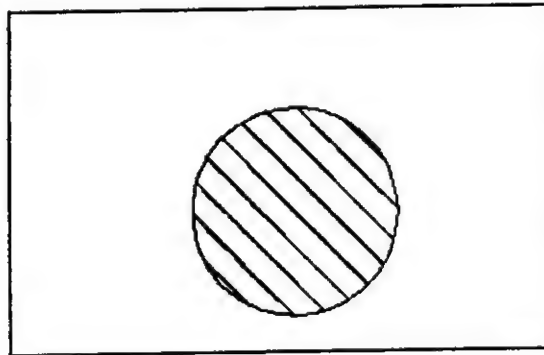


Figure 8-6. Example of SIF Area Patterning

### 8.3 ARC Command - Generate Arc

The SIF ARC command can define an IGDS circular arc. The ARC command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, all vertices must have x and y coordinates while 3-D SIF files must have x, y, and z coordinates. There are two forms of the SIF ARC command available. An arc can be defined by a center vertex, two vertices on the edge, and an arc direction or by three vertices on the edge of the arc. Both of these forms can also have a transformation matrix which is useful in engineering applications. The transformation matrix in a 2-D arc definition requires four terms while a 3-D arc definition requires nine terms.

NOTE: The transformation matrix for the SIF ARC command reflects pure rotation and should not be used as a scaling matrix. The range of the terms of the transformation matrix are from -1. to 1.

The transformation matrix should always be input as a row major matrix as follows:

$$\begin{bmatrix} t11 & t12 & t13 \\ t21 & t22 & t23 \\ t31 & t32 & t33 \end{bmatrix}$$

The SIF ARC command generates an IGDS circular arc element and the vertex range is from -2147483648 to 2147483647. All vertex coordinates should be in UORs (units of resolution). When using the SIF arc definition of center vertex and two vertices on the edge, the first vertex on the edge defines the beginning point of the arc, the center vertex defines the center of curvature of the arc in conjunction with the first vertex, the second vertex denotes the sweep angle of the arc, and the direction indicates whether this sweep angle is measured clockwise or counterclockwise from the first vertex. If using the three vertices on the edge definition, the first vertex defines the beginning point of the arc, the second vertex defines a point through which the arc passes and the third vertex defines the ending point of the arc. Since a SIF arc definition is an open element, it can also be part of a complex string definition.

In the following subsections, xc,yc represents the arc center vertex and xn,yn represents a vertex on the arc.

### B.3.1 ARC Command - ASCII Form

The SIF ASCII form of the ARC command is indicated by a command type of ARC followed by a slash (/). The slash can then be followed by one of the three different SIF ASCII forms that follow:

- a) ARC/CC, CE=xc,yc, P1=x1,y1,P2=x2,y2, MA=t11,t21,t12,t22
- b) ARC/CL, CE=xc,yc, P1=x1,y1,P2=x2,y2, MA=t11,t21,t12,t22
- c) ARC/P1=x1,y1,P2=x2,y2,P3=x3,y3, MA=t11,t21,t12,t22

In the SIF ASCII arc forms a and b, the CE keyword defines the coordinates of the center vertex of the circle while the P1 and P2 keywords define the two vertices on the arc. The arc is drawn from x1,y1 to x2,y2 depending on the CL and CC keywords. The CL keyword indicates a clockwise sweep from x1,y1 to x2,y2 while the CC keyword indicates a counterclockwise sweep from

x1,y1 to x2,y2. The MA keyword indicates the terms of the transformation matrix. The CE, P1, and P2 keywords are required while the CC, CL, and MA keywords are optional. If the CC or CL keywords are not specified, then a counterclockwise sweep from x1,y1 to x2,y2 is assumed. Refer to Figure 8-7.

In the SIF ASCII arc form c, the P1, P2, and P3 keywords indicate the three vertices on the arc and are required. The arc is drawn from "x1,y1" through "x2,y2" to "x3,y3". The MA keyword defining terms of the transformation matrix is optional.

### B.3.2 ARC Command - Binary Form

The binary form of the ARC command is indicated by a command type of 42. The words to follow field must contain the number of Integer\*4 words used to complete the command. The subtype field indicates which of the arc forms is being used while the value form is always set to 0. The following examples show the possible binary formats where 'stype' indicates which arc form is being used.

```

stype = 0 - arc with center, two vertices,
          counterclockwise sweep, no matrix

        1 - arc with center, two vertices, clockwise
          sweep, no matrix

        2 - arc with three vertices on edge, no
          matrix

status = 110 - arc with center, two vertices, counter
            clockwise sweep; with matrix

        101 - arc with center, two vertices, clockwise
            sweep; with matrix

        102 - arc with three vertices on edge; with
            matrix
  
```

NOTE: The values of the transformation matrix should be scaled by a factor of 10000 in order to preserve decimal accuracy.

42	*WORDS
STYPE	STATUS
XC	-
YC	-
X1	-
Y1	-
X2	-
Y2	-

42	*WORDS
STYPE	STATUS
XC	-
YC	-
X1	-
Y1	-
X2	-
Y2	-
T11	-10000.
T21	-10000.
T12	-10000.
T22	-10000.

42	*WORDS
STYPE	STATUS
X1	-
Y1	-
X2	-
Y2	-
X3	-
Y3	-

42	*WORDS
STYPE	STATUS
X1	-
Y1	-
X2	-
Y2	-
X3	-
Y3	-
T11	-10000.
T21	-10000.
T12	-10000.
T22	-10000.

Figure 8-7. Forms of the ARC Command - ASCII Form

### B.3.3 ARC Command - Interface Form

The interface form allows you to generate an ARC command by specifying the arc direction, center vertex, two vertices on the edge and a flag indicating the presence of a matrix. A second rotation interface form is provided to generate an arc by specifying three vertices on the edge and a flag indicating the presence of a transformation matrix. The subroutine formats the command and writes it to a SIF output file. The following examples describe the calling sequence required to generate a SIF ARC command:

```
CALL CMD42A(dir,center,point1,point2,value,mflag,matrix)
```

```
CALL CMD42B(point1,point2,point3,value,mflag,matrix)
```

dir        - Integer\*2 arc direction from vertex 1 to  
            vertex 2  
            0 - counterclockwise  
            1 - clockwise

center     - Integer\*4 array containing the center vertex  
            coordinate in UORs

point1    - Integer\*4 array containing the first vertex  
            coordinate in UORs

point2    - Integer\*4 array containing the second vertex  
            coordinate in UORs

point3    - Integer\*4 array containing the third vertex  
            coordinate in UORs

value      - Integer\*2 value reserved for later use

mflag     - Integer\*2 value indicating the presence of a  
            matrix  
            0 - no matrix present  
            1 - matrix present

matrix    - Real\*4 array containing the terms of the  
            transformation matrix

### B.3.4 ARC Command Restrictions

The restrictions for the ARC command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The ARC command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)

- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a BEGIN PARAGRAPH (PAR) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)

### 8.3.5 ARC Command Examples

This section presents examples of the SIF ARC command. The description of the example is presented first followed by the example. Both valid and invalid examples are presented and the ASCII form is used for all examples.

The following example places an arc with a 90-degree sweep in each of four quadrants of an x-y coordinate system. Figure 8-8 is a graphic representation of the following SIF ASCII commands.

```
ARC/CC,CE=0,0,P1=100,0,P2=0,100
ARC/CL,CE=0,0,P1=-100,0,P2=0,100
ARC/P1=100,0,P2=-70,-70,P3=0,-100
ARC/CL,CE=0,0,P1=100,0,P2=0,-100
```

The following example defines two 3-D semicircles in the X-Z plane. Figure 8-9 is a graphic representation of the following SIF ASCII commands.

```
ARC/CC,CE=100,100,100,P1=200,100,100,P2=0,100,100,MA=1.,0.
,0.,0.,1.,0.,0.,0.,1.
ARC/P1=0,100,100,P2=100,100,0,P3=200,100,100,MA=1.,0.
,0.,0.,0.,0.,0.,0.,1.
```

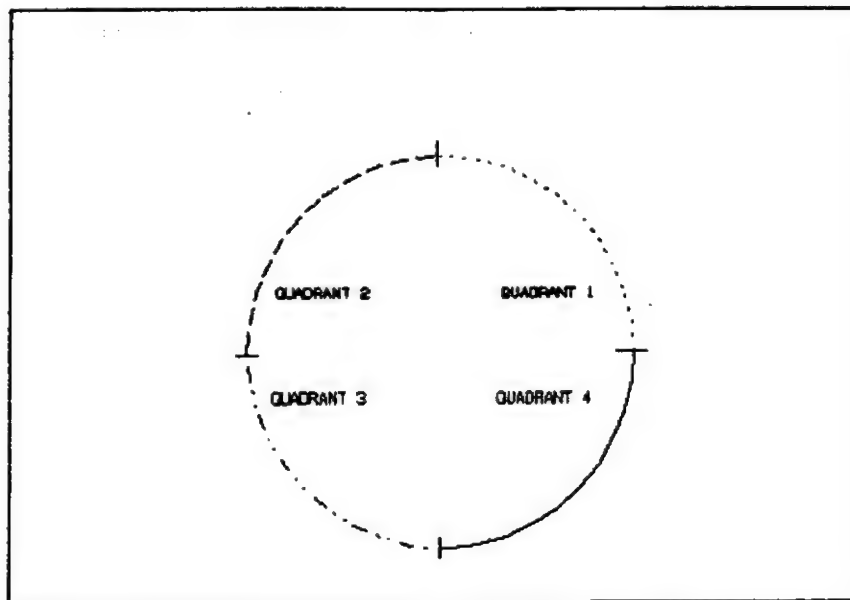


Figure 8-8. Example of SIF Circular Arcs

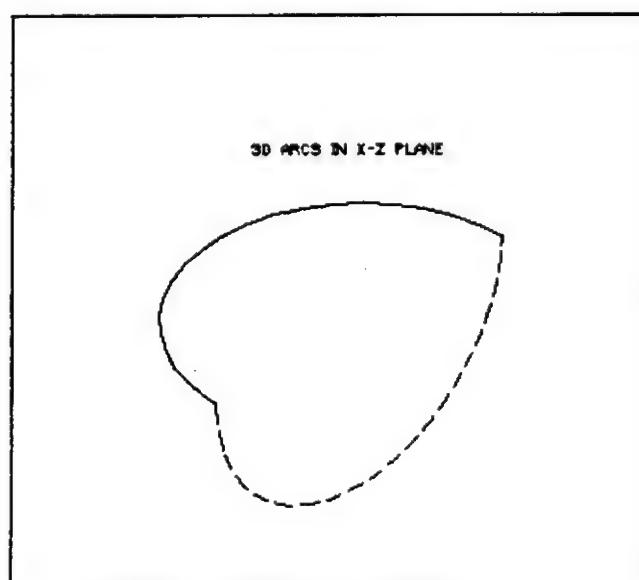


Figure 8-9. Example of SIF 3-D Arcs (ISO View)

The following example attempts to place an arc by center vertex and two vertices and it also shows the use of an arc in a SIF COMPLEX STRING command and in a SIF PATTERNING command. Figure 8-10 is a graphic representation of the following SIF ASCII commands.

```

BST/OP
LST/300,300,400,400
ARC/P1=400,400,P2=450,450,P3=500,400
LST/500,400,600,500,700,600
EST/
PTN/PS=0,PA=45.,ARCPAT
ARC/CL,CE=0,0,P1=100,0,P2=0,-100
PTN/OF

```

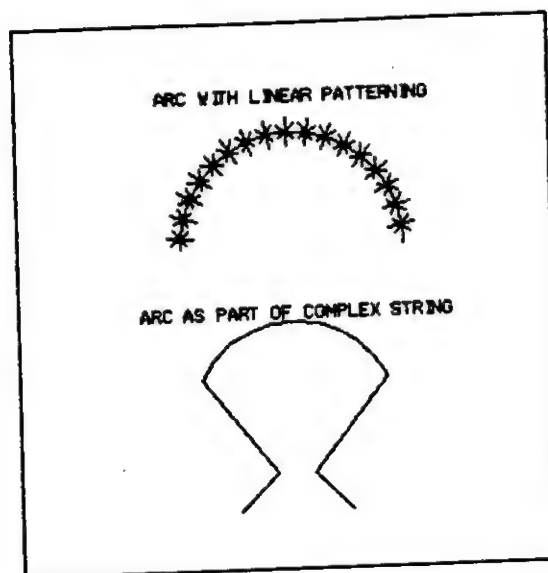


Figure 8-10. Example of SIF Arcs



#### B.4 SYMBOL Command - Generate Cell

The SIF SYMBOL command can define an IGDS cell. The SYMBOL command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates of a vertex are necessary while 3-D SIF files require x,y, and z coordinates. The SYMBOL command defines mirroring, the cell origin, the rotation matrix, and the cell name.

The mirroring field indicates whether the text line is to be mirrored about the x-axis or y-axis through the cell origin, or not mirrored at all. Mirroring can also be defined in the rotation matrix but if any text is included in the cell's definition, there is no guarantee that the text will be readable. By mirroring after placement, the text will be readable in top-to-bottom, left-to-right form.

The cell origin indicates the vertex at which the cell is to be placed. The vertex corresponds directly to a vertex defined when the cell was created and placed in a cell library.

The rotation matrix indicates the scale, rotation, and mirroring to be applied when the cell is placed. If not specified, no scale, rotation, or mirroring is applied. The matrix is computed as follows:

$$\begin{bmatrix} t11 & t12 \\ t21 & t22 \end{bmatrix} \text{ where}$$

$$\begin{aligned} t11 &= MY * XSC * \cos(\text{rot}) \\ t21 &= MX * XSC * \sin(\text{rot}) \\ t12 &= MY * YSC * -\sin(\text{rot}) \\ t22 &= MX * YSC * \cos(\text{rot}) \end{aligned}$$

where

MX - indicator for mirroring about the x-axis  
(1-no, -1-yes)

MY - indicator for mirroring about the y-axis  
(1-no, -1-yes)

rot - Rotation angle

XSC - X scale factor

YSC - Y scale factor

The cell name indicates the name given to the cell when it was created and placed in the cell library and is required. SIF allows cell names of 1 to 12 characters. If more than six characters are specified, only the first six are used to locate the cell in the cell library.

If a 3-D SIF file is being generated, a Z-coordinate must be added to the cell origin and the rotation matrix must contain nine terms.

#### 8.4.1 SYMBOL Command - ASCII Form

The ASCII form of the SYMBOL command is indicated by a command type of SYM followed by a slash (/). The slash is followed by one of the SIF ASCII forms that follow:

```
SYM/MX,OR=xo,yo,MA=t11,t21,t12,t22,celnam
SYM/MY,OR=xo,yo,MA=t11,t21,t12,t22,celnam
```

The MX or MY keyword indicates that the cell is to be mirrored after placement and is optional. MX indicates mirroring about the x-axis and MY indicates mirroring about the y-axis. Mirroring always occurs about the cell origin. If MX or MY is not specified, no mirroring occurs after placement.

The OR keyword indicates the cell origin in UORs and is required. You are required to know where the cell origin was defined when the cell was created and placed in the cell library.

The MA keyword indicates the rotation matrix to be used when the cell is placed and is optional. The MA keyword can be omitted if no matrix is desired or if the MATRIX command was used to generate the matrix. The MATRIX command is described in Section 4.1.

The "celnam" field indicates the cell name and is required.

#### 8.4.2 SYMBOL Command - Binary Form

The binary form of the SYMBOL command is indicated by a command type of 43. The words to follow field must contain the number of Integer\*4 used to complete the command. The subtype field indicates whether the cell is 2-D or 3-D and has a matrix. The value field indicates mirroring.

43	*WORDS
ST/PE	M/IR
XO	
YO	
SYMBOL NAME	

43	*WORDS
ST/PE	M/IR
XO	
YO	
T11	*10000.
T21	*10000.
T12	*10000.
T22	*10000.
SYMBOL NAME	

stype - 0 - 2-D symbol with no matrix  
           1 - 3-D symbol with no matrix  
           2 - 2-D symbol with matrix  
           3 - 3-D symbol with matrix

mir - 0 - no mirroring  
       1 - mirror about x-axis  
       2 - mirror about y-axis

#### 8.4.3 SYMBOL Command - Interface Form

The interface form allows you to generate a SYMBOL command by specifying the symbol type, mirroring, the cell origin, rotation matrix, the number of characters in the cell name, and the cell name. The subroutine formats the command and writes it to a SIF output file. The following example describes the calling sequence required to generate a SIF SYMBOL command:

CALL CMD43(stype,mir,origin,matrix,nc,celnam)

stype - Integer\*2 symbol type  
           0 - 2-D symbol with no matrix  
           1 - 3-D symbol with no matrix  
           2 - 2-D symbol with matrix  
           3 - 3-D symbol with matrix

mir - Integer\*2 mirror indicator  
       0 - no mirror  
       1 - mirror about x-axis through cell origin  
       2 - mirror about y-axis through cell origin

origin - Integer\*4 array containing the vertex for the cell origin in UORs

matrix - Real\*4 array containing the rotation matrix

nc - Integer\*2 number of characters in cell name

celnam - byte array containing cell name

The interface utility subroutine SIFMAT is provided to calculate the rotation matrix for the cell.

#### 8.4.4 SYMBOL Command Restrictions

The restrictions for the SYMBOL command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command.

The SYMBOL command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE LINKAGE ON (MID/ON) and a MULTIPLE LINKAGE OFF command (MID/OFF)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a BEGIN PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP)
- o between a BEGIN PATTERNING (PTN/) and a STOP PATTERNING command (PTN//OF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o before a CONTINUE command (CON/)

#### 8.4.5 SYMBOL Command Examples

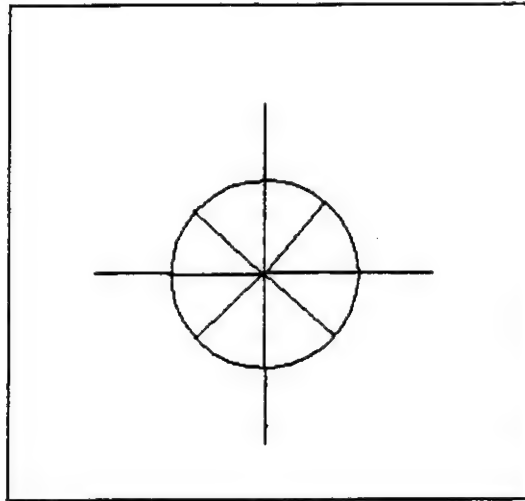
This section presents examples of the SIF SYMBOL command. The description of the example is presented first followed by the example. Both valid and invalid examples are presented and the ASCII form of the command is used for all examples.

The following example places a 2-D symbol named FH that has been scaled to twice its original size. Figure 8-11 is a graphic representation of the SIF ASCII command.

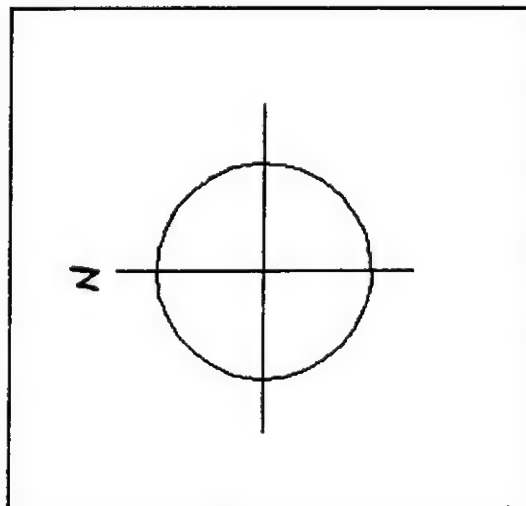
```
SYM/OR=200,500,MA=2.,0,0.,2.,FH
```

The following example places a 3-D symbol named XHAIR which is rotated 90 degrees. Figure 8-12 is a graphic representation of the SIF ASCII command.

```
SYM/OR=100,100,0,MA=0.,1.,0.,-1.,0.,0.,0.,0.,1.,XHAIR
```



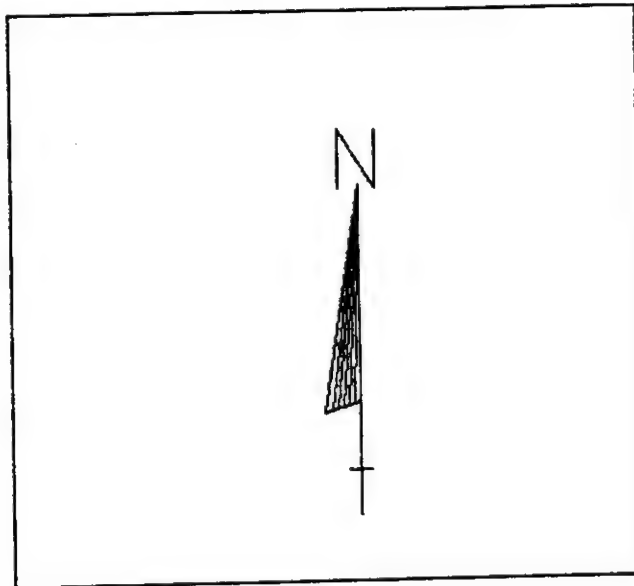
**Figure 8-11. Example of SIF Scaled Symbol**



**Figure 8-12. Example of SIF Rotated Symbol**

The following example places a 2-D symbol named NARR which is to be mirrored about the y-axis through the origin:

SYM/MY,OR=-200,100,NARR



The following example attempts to place a 2-D symbol named CROSSHAIR which is neither mirrored, scaled, or rotated:

SYM/OR=500,-500,CROSSHAIR

NOTE: The symbol name CROSSHAIR is more than six characters long and if the cell library does not contain a cell named CROSSH the symbol is not placed.

#### 8.5 INCLUDE TEXT Command - Enter Data Fill-In

The SIF INCLUDE TEXT command is supported in IGDS as an enter data fill-in. If INCLUDE TEXT commands are used, they must appear immediately after a SIF GENERATE SYMBOL, GENERATE TEXT LINE, or CLOSE PARAGRAPH command. The INCLUDE TEXT command defines the line number, delimiter character, and text which contains the enter data field(s).

The line number defines the text line number for which the INCLUDE TEXT command applies. The line number has a range of 1 to 50. If the command appears after a GENERATE TEXT LINE command, the line number must always be 1. After a GENERATE SYMBOL or CLOSE PARAGRAPH command, the line number is the actual text string in the cell or text node. For a cell, you must know the order in which the text strings were defined. All text strings are counted whether they contain enter data fields or not.

The delimiter character separates the enter data fields for text lines which contain multiple enter data fields. Any ASCII character which is not used for the enter data field is valid. If only one enter data field exists in the text line, the delimiter character is not required.

The text which contains the enter data fields is an ASCII string with the delimiter character separating the enter data field(s) on the text line. If the delimiter character is not specified, the entire text field is assumed to be the first enter data field on the line. Two consecutive delimiters indicate an enter data field to be skipped.

#### B. 5.1 INCLUDE TEXT Command - ASCII Form

The ASCII form of the INCLUDE TEXT command is indicated by a command type of INC followed by a slash (/). The slash is followed by the SIF ASCII forms that follow:

INC/LI=line,DE=delim,text

The LI keyword indicates line and is the line number containing the enter data field. The LI keyword is required following a SIF SYMBOL or PARAGRAPH command but is optional if following a SIF TEXT command. If omitted, "line" is assumed to be one (1).

The DE keyword indicates delim and is the delimiter character separating enter data field in the text field. It is optional. If omitted, the entire text field is assumed to be in the first enter data field on the line.

The text field defines the text to be placed in the enter data field(s) on the given line and includes any required delimiters.

#### B. 5.2 INCLUDE TEXT Command - Binary Form

The binary form of the INCLUDE TEXT command is indicated by a command type of 44. The words to follow field must contain the number of Integer\*4 words used to complete the command. The subtype field gives the line number, line of the symbol, or text node containing the enter data field. For text, this value is always one (1). The value field indicates the delimiter used. If delim is 0 or blank, the entire text field is assumed to be the first enter data field on the line.

44	*WORDS
LINE	DELIM
TEXT	

### 8.5.3 INCLUDE TEXT Command - Interface Form

The interface form allows you to generate an INCLUDE TEXT command by specifying text line and field numbers, a delimiter, the number of characters in the text array, and a byte array containing text to be placed in the enter data fields. The subroutine formats this information and writes it to a SIF output file. The following example describes the calling sequence required to generate a SIF INCLUDE TEXT command:

```
CALL CMD44(line,field,delim,nc,text)
```

line - Integer\*2 text line number

field - Integer\*2 field number within the text line. A field number of 0 indicates that the text array contains delimiters. A field number greater than 0 indicates that the text array is for only one field and the interface library will insert delimiters where necessary.

delim - One byte delimiter to be used to separate fields in the text array. If 0 or blank, the entire text array is assumed to be the first enter data field. If "field" is greater than 0, a delimiter character must be specified.

nc - Integer\*2 number of characters in the text array.

text - Byte array containing text to be placed in the enter data fields.

### 8.5.4 INCLUDE TEXT Command Restrictions

The restrictions for the INCLUDE TEXT command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The INCLUDE TEXT command may appear in the SIF file only in the following conditions:

- o immediately after a GENERATE TEXT command (TXT/)



- o immediately after a GENERATE SYMBOL command (SYM/)
- o immediately after an END SYMBOL command (ESY/)
- o immediately after a CLOSE PARAGRAPH command (CLP/)

#### 8.5.5 INCLUDE TEXT Command Examples

This section presents examples of the SIF INCLUDE TEXT command. The description of the example is represented first followed by the example. Both valid and invalid examples are presented and the ASCII form of the command is used in all examples.

The following example places a symbol named "FLAG". The text values of 12-DEC-58 are placed in the third field of the second text string in symbol FLAG.

```
SYM/OR=200,150,FLAG
INC/LI=2,DE=?,???12-DEC-58?
```

The following example places a text node with three text strings each of which contain enter data fields. INCLUDE TEXT commands edit each text string according to the specified fields and delimiters.

```
TPC/CO=3-D,SP=25,NU=3,JU=3
PAR/OR=-1000,20,AN=30.,TH=210,TW=110
PLN/NOW-----TIME
PLN/FOR ALL----WOMEN TO
PLN/COME--THE---OF---COUNTRY
CLP/
INC/LI=1,DE=*,*IS THE*
INC/LI=2,DE=?,?GOOD?
INC/LI=3,DE=**,**AID*
```

At a graphics terminal, you see the following figures. Note that Figure A has display enter data field turned off while Figure B has the display turned on:

```
NOW-----TIME
FOR ALL ---- WOMEN TO
COME -- THE --- OF --- COUNTRY
```

Figure A

```
NOW IS THE TIME
FOR ALL GOOD WOMEN TO
COME THE AID COUNTRY
```

Figure B

The following example places a text string with two enter data fields. The INCLUDE TEXT command which follows would be illegal because the delimiter appears within the text to be placed in the field:

```
TXT/OR=1000,500,AN=20,-----WHO GOES -----
INC/DE=!,!STOP!!THERE?!
```

The meaning of the INCLUDE TEXT command in this situation is that text string has three enter data fields when in actuality you are trying to generate a text string that looks like the following: STOP! WHO GOES THERE?

## 8.6 CURVE Command - Generate Curve

The SIF CURVE command defines a curve containing no fewer than three vertices. The CURVE command is available in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates must be specified for each vertex while a 3-D SIF file requires x, y and z coordinates for each vertex. The SIF CURVE command can be continued with a SIF CONTINUE command or the SIF ASCII continuation line (four blanks at the beginning of the next record). The SIF CURVE command allows an unlimited number of vertices but any curve that has more than 101 vertices is broken in curves of 101 vertices or less. The formats for the SIF ASCII and SIF binary forms of the curve are presented in the following sections.

### 8.6.1 CURVE Command - ASCII Form

The ASCII form of the CURVE command is indicated by a command type of CUR followed by a slash (/). The slash is then followed by the coordinates of the vertices. The following are examples of SIF ASCII CURVE commands:

```
CUR/x1,y1,x2,y2,y3,y3,x4,y4,x5,y5
CUR/x1,y1,z1,x2,y2,z2,x3,y3,z3
```

The vertices must be in the range of -2147483648 to 2147483647 and they must be given in UORs.

### 8.6.2 CURVE Command - Binary Form

The binary form of the CURVE command is indicated by a command type of 45 in the command header. The words to follow field must contain the number of Integer\*4 words to follow in the definition where each coordinate of a vertex occupies one Integer\*4 word. The subtype field and value fields are not used.

45	*WORDS
0	0
X1	
Y1	
X2	
Y2	
...	
XN	
YN	

### 8.6.3 CURVE Command - Interface Form

The interface form allows you to generate a CURVE command by specifying the number of vertices and the coordinates of the vertices. The subroutine formats the command and writes the command to the output SIF file. The interface library automatically generates SIF CONTINUE commands where required or you can continue the curve with a SIF CONTINUE command.

The most common form of the calling sequence, though, is as follows:

```
CALL CMD45(nv,vert)
```

nv - Integer\*2 number of vertices in the vert array

vert - Integer\*4 array containing the vertices of the curve in UORs.

The interface utility subroutine SIFPTS is provided to allow you to specify the vertices for one curve with multiple calls. SIFPTS is useful if you do not know the number of vertices required for the curve or do not have enough buffer space to store all vertices for the curve. If SIFPTS is used, the calling sequence for a SIF CURVE command must initially be as follows:

```
CALL CMD45(0,0)
```

### 8.6.4 CURVE Command Restrictions

The restrictions for the CURVE command are described in this section. For a complete description of the conflicting commands, refer to the section which describes that command. The CURVE command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o immediately before an INCLUDE TEXT command (INC/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)

### 8.6.5 CURVE Command Examples

This section presents examples of the SIF CURVE command. Each example is preceded by a brief description. The SIF ASCII form is used for all examples.

The following example places 2-D curves through the given vertices. Figure 8-13 is a graphic example of the SIF ASCII commands.

```
CUR/100,100,200,0,300,100,500,200,550,150
LAC/LS=2
CUR/100,300,200,200,300,300,500,500,550,350
LAC/LS=4
CUR/100,500,200,400,300,500,500,700,550,550
```

The following example places 3-D curves through the given vertices. Figure 8-14 is a graphic example of the SIF ASCII commands.

```
CUR/100,100,100,200,100,0,300,100,100,500,100,
    200,550,100,150
LAC/LS=2
CUR/100,100,200,200,100,100,300,100,200,500,100,
    300,550,100,250
LAC/LS=4
CUR/100,100,300,200,100,200,300,100,300,500,100,
    400,550,100,350
LAC/LS=6
CUR/100,100,400,200,100,300,300,100,400,500,100,
    500,550,100,450
```

The following example attempts to place a 2-D curve:

```
CUR/-300,0,-200,-100
```

This is an invalid definition because it takes at least three vertices to define a curve.

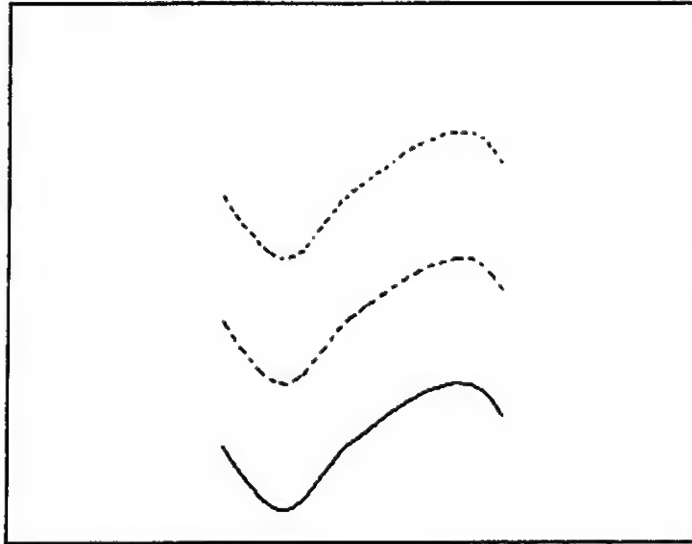


Figure 8-13. Example of SIF 2-D Curves

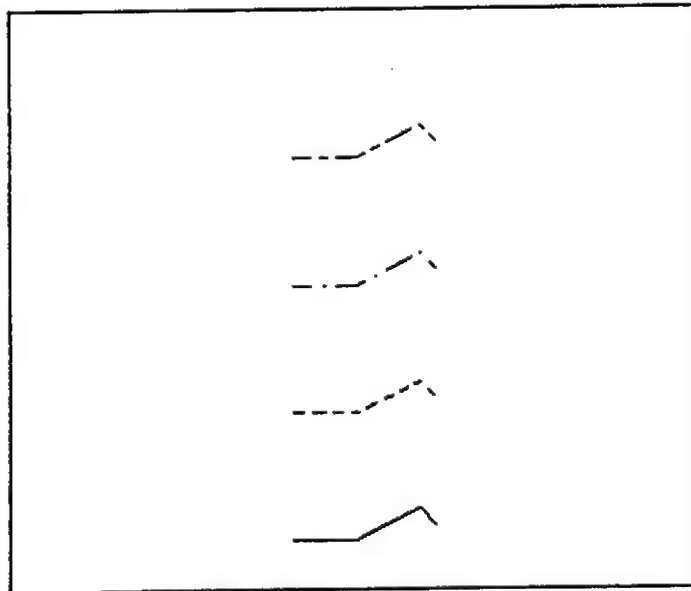


Figure 8-14. Example of SIF 3-D Curves

## 8.7 ELLIPSE Command - Generate Ellipse

The SIF ELLIPSE command defines an IGDS ellipse. The ELLIPSE command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates of a vertex must be specified while an additional z coordinate is necessary in 3-D files. The SIF ELLIPSE command can also have a transformation matrix which defines rotation in the x-y plane for 2-D ellipses and rotation in space for 3-D ellipses. The transformation matrix is always row major and must contain four terms for 2-D files and nine terms for 3-D files. It is important to note that the transformation matrix is for rotation and not for scaling purposes.

The IGDS ellipse element generated from the SIF ELLIPSE command can have a vertex range from -2147483648 to 2147483647 and the vertices must always be given in UORs. The SIF ellipse is defined in both 2-D and 3-D files with a center vertex, a vertex lying on the primary axis and a vertex lying on the secondary axis. These two vertices along with the center vertex are used to establish magnitudes of the primary and secondary axes of the ellipse. If no transformation matrix is given, the primary axis is used to establish rotation. Since a SIF ellipse is supported in IGDS as a closed element, it can be area patterned using the SIF PATTERNING command.

### 8.7.1 ELLIPSE Command - ASCII Form

The ASCII form of the ELLIPSE command is indicated by a command type of ELL followed by a slash (/). The slash is then followed by the ellipse center vertex, a vertex on the primary axis, a vertex on the secondary axis, and a transformation matrix, if desired. Examples of the SIF ASCII ELLIPSE command are as follows:

```
ELL/CE=x1, y1, P1=x2, y2, P2=x3, y3, MA=t11, t21, t12, t22  
ELL/HO, CE=x1, y1, z1, P1=x2, y2, z2, P2=x3, y3, z3, MA=t11, t21, t31,  
t12, t22, t23, t13, t23, t33
```

The HO keyword is optional and indicates that the element is a hole and not a solid. If not specified, the default of solid is taken. The CE keyword is required and indicates the center of the ellipse. The P1 keyword is required and indicates a vertex on the ellipse edge that lies the length of the primary axis from the center vertex. The P2 keyword is required and represents a vertex on the ellipse that lies the length of the secondary axis from the center vertex. The MA keyword is optional and indicates the terms of the row major transformation matrix. The matrix should have four terms for the 2-D ellipse and nine terms for the 3-D ellipse. The terms of the matrix are in a range from -1. to 1.

### 8.7.2 ELLIPSE Command - Binary Form

The binary form of the ELLIPSE command is indicated by a command type of 47 in the command header. The words to follow field must contain the number of Integer\*4 words which are used to complete the definition. The subtype field indicates whether a matrix is present while the value field indicates the status of the ellipse as either a hole or a solid.

47	*WORDS
STYPE	STATUS
XC	-
YC	-
X1	-
Y1	-
X2	-
Y2	-

47	*WORDS
STYPE	STATUS
XC	-
YC	-
X1	-
Y1	-
X2	-
Y2	-
T11 *10000.	-
T21 *10000.	-
T12 *10000.	-
T22 *10000.	-

stype - 0 - no matrix present  
100 - matrix present

status - 0 - solid  
1 - hole

NOTE: The binary values of the transformation matrix are scaled by a factor of 10000. in order to preserve decimal accuracy.

### 8.7.3 ELLIPSE Command - Interface Form

The interface form allows you to generate an ELLIPSE command by specifying a status, center vertex, two points on the ellipse, and a matrix flag indicating the presence of a transformation matrix. The subroutine formats the command and writes it to a SIF output file. The following example

describes the calling sequence that is used when generating a SIF ELLIPSE command using the interface form:

CALL CMD47(center, point1, point2, status, mflag, matrix)

center - Integer\*4 array containing coordinate of center vertex of circle in UORs

point1 - Integer\*4 array containing coordinate of a point the distance from the origin of the primary axis

point2 - Integer\*4 array containing coordinate of a point the distance from the origin of the secondary axis

status - Integer\*2 element status  
0 - solid  
1 - hole

mflag - Integer\*2 flag indicating whether a matrix is given (0-no, 1=yes)

matrix - Real\*4 array containing the terms of the transformation matrix

#### B.7.4 ELLIPSE Command Restrictions

The restrictions of the ELLIPSE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The ellipse may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o before a CONTINUE command (CON/)



### B.7.5 ELLIPSE Command Examples

This section presents examples of the SIF ELLIPSE command. The description is presented first followed by the example. Both valid and invalid examples are given and the ASCII form of the command is always used.

The following example places two ellipses at the same origin but with their primary axis lying at different rotations. Figure 8-15 is a graphic representation of the following SIF ASCII commands:

```
OVR/10
LAC/LS=1
ELL/CE=100, 100, P1=100, 300, P2=0, 100, MA=1., 0., 0., 1.
LAC/LS=2
ELL/CE=100, 100, P1=300, 100, P2=100, 200
```

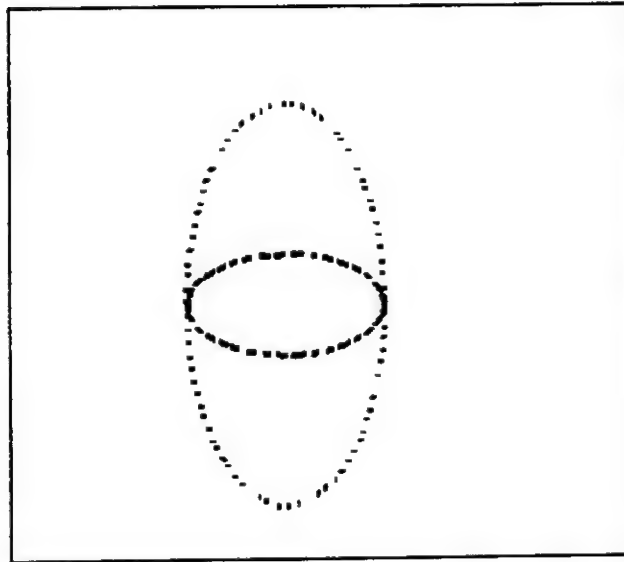


Figure 8-15. Example of SIF Ellipses

The following example defines two 3-D ellipses in ISO view, one as a solid and one as a hole. Figure 8-16 is a graphic representation of the following SIF ASCII commands.

```
ELL/CE=100, 0, 100, P1=200, 0, 100, P2=100, 0, 200
ELL/HO, CE=0, 100, 200, P1=0, 200, 200, P2=0, 100, 100, MA=. 707, 0.,
    0., 0., . 707, 0., 0., 0., 1.
```

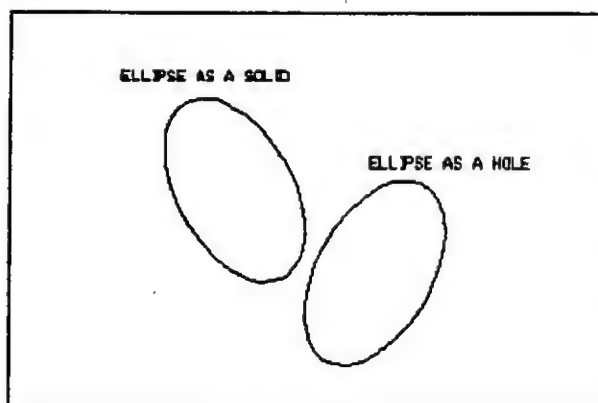


Figure 8-16. Example of SIF 3-D Ellipses (ISO View)

The following example attempts to place ellipses in all the wrong places. Each attempt to place an ellipse is illegal.

```
IDE/AS=1, ID=2, CO=0, KE=0
ELL/CE=0, 0, P1=100, 0, P2=0, 50
ASV/ID=1, DE=!, !1!HOUSE!2!RIVER CITY!
MID/ON
IDE/AS=4, ID=24, ORF
IDE/AS=4, ID=24, CELL
ELL/CE=500, 500, P1=700, 500, P2=500, 600
IDE/AS=4, ID=24, HERE
MID/OF
BST/
LST/100, 100, 200, 200, 400, 100
EST/
```

## 8.8 ELLIPTICAL ARC Command - Generate Partial Ellipse

The SIF ELLIPTICAL ARC command defines an IGDS partial ellipse element. The ELLIPTICAL ARC command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates of a vertex must be specified while an additional z coordinate is necessary in 3-D files. The SIF ELLIPTICAL ARC command can also have a transformation matrix which defines rotation in the x-y plane for 2-D elliptical arc and rotation in space for 3-D elliptical arcs. The transformation matrix is always row major and must contain four terms for 2-D files and nine terms for 3-D files.

NOTE: The transformation matrix is purely for rotation and not for scaling purposes.

The IGDS partial ellipse element generated from the SIF ELLIPTICAL ARC command can have a vertex range from -2147483648 to 2147483647 and the vertices must always be given in UORs. The SIF elliptical arc is defined with a center vertex, a vertex lying on the arc edge at the primary axis, a vertex lying on the arc edge at the secondary axis, a start angle measured counterclockwise from the primary axis, and a sweep angle. The points lying on the edge of the arc are used to determine rotation and axes magnitude. The start and sweep angles determine the actual partial ellipse to be placed. The start and sweep angles are floating point values with five decimal digit accuracy.

### 8.8.1 ELLIPTICAL ARC Command - ASCII Form

The ASCII form of the ELLIPTICAL ARC command is indicated by a command type of EAR followed by a slash (/). The slash is then followed by the elliptical arc center vertex, a vertex on the arc edge at the primary axis, a vertex on the arc edge at the secondary axis, a start angle, a sweep angle, and a transformation matrix. An example of the SIF ASCII ELLIPTICAL ARC command is as follows:

```
EAR/CE=x0, y0, P1=x1, y1, P2=x1, y1, ST=sta, SW=swa,  
MA=t11, t21, t12, t22
```

The CE keyword is required and indicates the center of the arc. The P1 keyword is required and indicates a vertex on the arc edge that lies the length of the primary axis from the center vertex. The P2 keyword is required and indicates a vertex on the arc edge that lies the length of the secondary axis from the center vertex. The ST keyword indicates the start angle and is the angle measured counterclockwise from the primary axis of the elliptical arc. The ST keyword is required. The SW keyword is required and is the angle measured counterclockwise from the start angle. The MA keyword is

optional and indicates the terms of the row major transformation matrix. The matrix should have four terms for the 2-D elliptical arc and nine terms for the 3-D elliptical arc. The range of the terms of the matrix is from -1. to 1.

#### 8.8.2 ELLIPTICAL ARC Command - Binary Form

The binary form of the ELLIPTICAL ARC command is indicated by a command type of 48 in the command header. The words to follow field must contain the number of Integer\*4 words which are necessary to complete the definition. The subtype field indicates whether a transformation matrix is present. Refer to Figure 8-17.

stype    -    0    no matrix present  
          - 100   matrix present

NOTE: The binary values of the transformation matrix are scaled by a factor of 10000 in order to preserve decimal accuracy.

#### 8.8.3 ELLIPTICAL ARC Command - Interface Form

The interface form allows you to generate an ELLIPTICAL ARC command by specifying a center vertex, two points on the ellipse, a start and a sweep angle, a matrix flag indicating the presence of a transformation matrix, and a matrix. The subroutine formats the command and writes it to a SIF output file. The following example describes the calling sequence that is used when generating a SIF ELLIPTICAL ARC command using the interface form:

```
CALL CMD48(center, point1, point2, sta, swa, value, mflag,  
          matrix)
```

center    - Integer\*4 array containing elliptical arc origin in UORs

point1    - Integer\*4 array containing a point the distance from the origin of the primary axis

point2    - Integer\*4 array containing a point the distance from the origin of the secondary axis

sta        - Real\*4 start angle measured counterclockwise from the primary axis

4B	*WORDS
STYPE	0
XC	
YC	
X1	
Y1	
X2	
Y2	
STA	*10000.
SWA	*10000.

4B	*WORDS
STYPE	0
XC	
YC	
X1	
Y1	
X2	
Y2	
STA	*10000.
SWA	*10000.
T11	*10000.
T21	*10000.
T12	*10000.
T22	*10000.

Figure B-17. ELLIPTICAL ARC Command - Binary Form

swa        - Real\*4 sweep angle of arc  
 value      - Integer\*2 value reserved for future use  
 mflag      - Integer\*2 flag indicating whether a matrix is  
                  given (0-no matrix; 1-matrix)  
 matrix     - Real\*4 array containing the transformation  
                  matrix

#### 8.8.4 ELLIPTICAL ARC Command Restrictions

The restrictions of the ELLIPTICAL ARC command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The elliptical arc may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF).
- o immediately before an INCLUDE TEXT command (INC)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)

#### 8.8.5 Elliptical ARC Command Examples

This section presents examples of the SIF ELLIPTICAL ARC command. The description is presented first followed by the example. Both valid and invalid examples are given and the ASCII form of the command is always used.

The following example places two elliptical arcs on level 13 at the same origin but at start angles that differ by 180 degrees. Figure 8-18 is a graphic representation of the following SIF ASCII commands.

```

DVR/13
EAR/CE=100,100,P1=200,100,P2=100,150,ST=270,SW=180
LAC/LS=3
EAR/CE=100,100,P1=200,100,P2=100,150,ST=90,SW=180

```

The following example places a 3-D elliptical arc which is rotated using the transformation matrix. Figure 8-19 is a graphic representation of the following SIF ASCII commands.

```

EAR/CE=0,0,0,P1=500,0,0,P2=0,100,0,ST=45,SW=90,MA=.707,0.,
0.,0.,0.,.707,0.,0.,0.,1.

```

The following example places an elliptical arc as part of a complex element definition. Figure 8-20 is a graphic representation of the following SIF ASCII commands.

```
DVR/50
LAC/LS=1,LT=-3
BST/
LST/100,100,200,200,300,300
EAR/CE=300,100,P1=300,300,P2=250,100,ST=0.,SW=180
LST/300,-100,200,0,100,100
EST/
```

The following is an illegal definition because the secondary axis has no magnitude:

```
EAR/CE=1000,500,P1=1000,1000,P2=1000,500,ST=90,SW=45
```



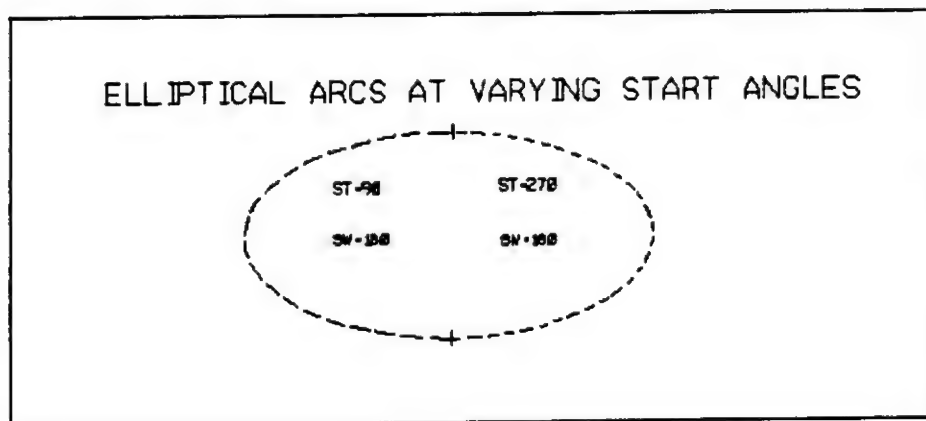


Figure 8-18. Example of SIF 2-D Elliptical Arcs

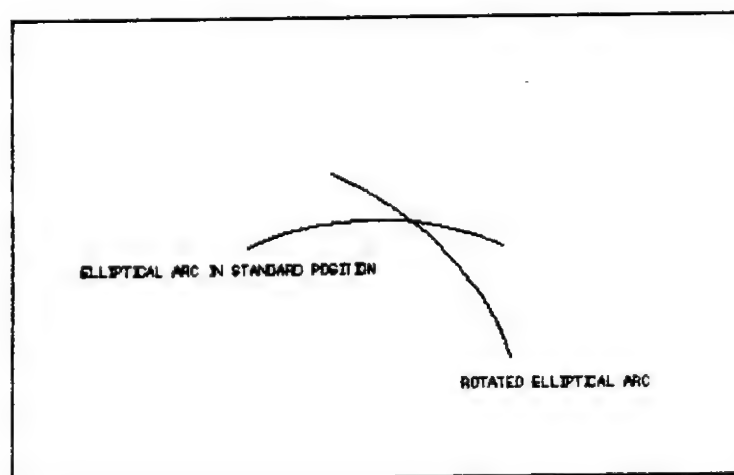


Figure 8-19. Example of SIF 3-D Elliptical Arcs

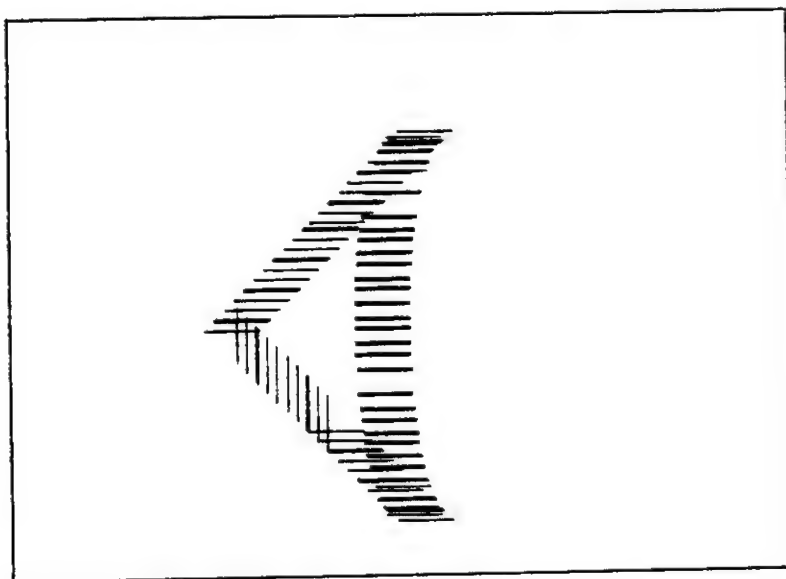


Figure 8-20. Example of SIF Elliptical Arc in Complex String

## 8.9 CONIC Command - Generate Conic

The SIF CONIC command can define an IGDS conic element containing no fewer than three vertices. The CONIC command may be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates must be specified for each vertex while a 3-D SIF file requires x, y, and z coordinates for each vertex. The SIF CONIC command can be continued with a SIF CONTINUE command or the SIF ASCII continuation line (four blanks at the beginning of the next record). The SIF CONIC command allows an unlimited number of vertices but any conic that has more than 101 vertices will be broken in conics of 101 vertices or less. The formats for the SIF ASCII and SIF binary forms of the conic are presented in the following sections.

### 8.9.1 CONIC Command - ASCII Form

The ASCII form of the CONIC command is indicated by a command type of CNC followed by a slash (/). The slash is then followed by the coordinates of the vertices. The following are examples of SIF ASCII CONIC commands:

```
CNC/x1, y1, x2, y2, x3, y3, x4, y4, x5, y5
CNC/x1, y1, z1, x2, y2, z2, x3, y3, z3
```

The vertices must be in the range of -2147483648 to 2147483647 and they must be given in UORs.

### 8.9.2 CONIC Command - Binary Form

The binary form of the CONIC command is indicated by a command type of 49 in the command header. The words to follow field must contain the number of Integer\*4 words to follow in the definition where each coordinate of a vertex occupies one Integer\*4 word. The subtype field and value fields are not used.

49	*WORDS
0	0
X1	
Y1	
X2	
Y2	
XN	
YN	

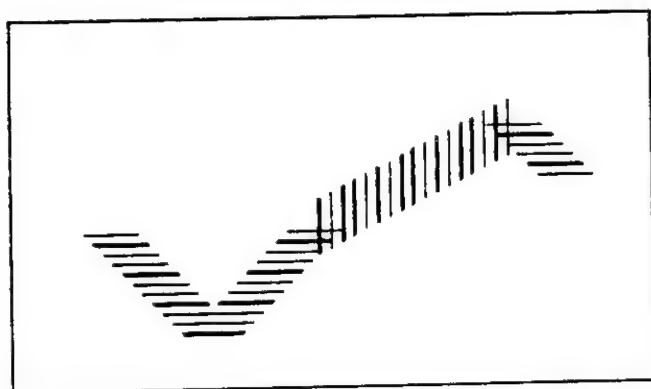


Figure 8-21. Example of SIF 2-D Conic

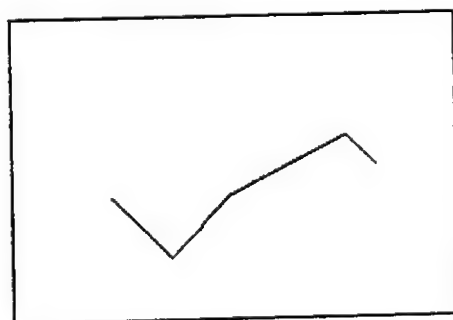


Figure 8-22. Example of SIF 3-D Conic

#### 8.9.5 CONIC Command Examples

This section presents examples of the SIF CONIC command. Each example is preceded by a brief description and the SIF ASCII form is used for all examples.

The following example places a 2-D conic through the given vertices. Figure 8-21 is a graphic representation of the following SIF ASCII command.

```
CNC/100,100,200,0,300,100,500,200,550,150
```

The following example places a 3-D conic through the given vertices. Figure 8-22 is a graphic representation of the following SIF ASCII command.

```
CNC/100,100,100,200,100,0,300,100,100,500,100,200,550,100,  
150
```

The following example attempts to place a 2-D conic:

```
CNC/-300,0,-200,-100
```

This would be an invalid definition because it takes at least three vertices to define a conic.

#### 8.10 BEGIN COMPLEX STRING Command - Generate Complex String or Complex Shape

The SIF BEGIN COMPLEX STRING command is supported in IGDS as a complex string or complex shape. If this command is used, the SIF END STRING command (see Section 8.11) must also be used to indicate the end of the complex string. The SIF commands which can appear in a complex string set are the OVERLAY, CLASSIFICATION, LINE/AREA CHARACTERISTICS (line style, line weight, and line color only), GENERATE LINE STRING (open line string only), GENERATE ARC, GENERATE ELLIPTICAL ARC, GENERATE CURVE, GENERATE CONIC, and CONTINUE commands. You are responsible for ensuring the connectivity of the internal elements of the complex string. For closed shapes, the first vertex of the first internal element must be equal to the last vertex of the last internal element.

#### 8.10.4 BEGIN COMPLEX STRING Command Restrictions

The restrictions of the BEGIN COMPLEX STRING command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN COMPLEX STRING command must always be in conjunction with the END COMPLEX STRING command and the pair may appear anywhere in the SIF file except for the following conditions.

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)

There are also restrictions on the commands that can appear in a complex string set. The SIF commands that can appear in the set are the OVERLAY, CLASSIFICATION, LINE/AREA, CHARACTERISTICS (line style, line weight, and line color only), GENERATE LINE STRING (open line string only), GENERATE ARC, GENERATE ELLIPTICAL ARC, GENERATE CURVE, GENERATE CONIC, and CONTINUE commands.

#### 8.10.5 BEGIN COMPLEX STRING Command Examples

This section presents examples of the SIF BEGIN COMPLEX STRING command. A description is given first followed by the example itself. All examples are given in the ASCII form.

The following example places multiple linkage user data on a complex string composed of line strings and an arc. The complex string is on level 10 and has a line style of dash dot. Figure 8-23 is a graphic representation of the following SIF ASCII commands.

### 8.10.1 BEGIN COMPLEX STRING Command - ASCII Form

The ASCII form of the BEGIN COMPLEX STRING command is indicated by a command type of BST followed by a slash (/). The slash can then be followed by one of three keywords indicating the type of complex string being generated. Examples of the SIF ASCII BEGIN COMPLEX STRING command are as follows:

```
BST/OP
BST/SO
BST/HO
```

The OP keyword indicates an IGDS complex string and is optional. If omitted, the default is a complex string.

The SO keyword indicates an IGDS complex shape representing a solid and is required.

The HO keyword indicates an IGDS complex shape representing a hole and is required.

### 8.10.2 BEGIN COMPLEX STRING Command - Binary Form

The binary form of the BEGIN COMPLEX STRING command is indicated by a command type of 50 in the command header. The words to follow field is always 0. The subtype field indicates that the type of complex element and the value field is always 0.

50	0
LTYPE	0

```
ltype = 1 - complex string
        2 - complex shape representing a solid
        3 - complex shape representing a hole
```

### 8.10.3 BEGIN COMPLEX STRING Command - Interface Form

The interface form allows you to generate a BEGIN COMPLEX STRING command by specifying the type of complex element desired. The subroutine formats the command and writes it to a SIF output file. The following example describes the calling sequence that is used when generating a SIF BEGIN COMPLEX STRING command using the interface form:

```
CALL CMD50(ltype)
```

```
ltype-Integer*2 complex string type
      1 - complex string
      2 - complex shape representing a solid
      3 - complex shape representing a hole
```

#### 8.11.1 END COMPLEX STRING Command - ASCII Form

The ASCII form of the END COMPLEX STRING command is indicated by a command type of EST followed by a slash (/).

EST/

#### 8.11.2 END COMPLEX STRING Command - Binary Form

51	0
0	0

The binary form of the END COMPLEX STRING command is indicated by a command type of 51 in the command header. All other fields are zero.

#### 8.11.3 END COMPLEX STRING Command - Interface Form

The interface form allows you to generate an END COMPLEX STRING command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD51

#### 8.11.4 END COMPLEX STRING Command Restrictions

See Section 8.10.4.

#### 8.11.5 END COMPLEX STRING Command Examples

See Section 8.10.5.

#### 8.12 BEGIN SYMBOL Command - Expanded Cell; Orphan Cell

The SIF BEGIN SYMBOL command can be used if you desire all simple elements of a cell to be included in the SIF file along with the SYMBOL command. If a BEGIN SYMBOL command is used, the SIF SYMBOL command must be used to indicate the end of the cell's simple elements. The SIF commands which can be included in a SYMBOL command set are the OVERLAY CLASSIFICATION, FONT, LINE/AREA CHARACTERISTICS (line style, line weight, and line color only), TEXT LINE CHARACTERISTICS, PARAGRAPH CHARACTERISTICS, and GRAPHIC ELEMENT GENERATION commands. If there is a cell library specified in the SIF environment file and the SYMBOL command in the symbol set is in that cell library, the cell is placed using the definition from the cell library. All simple elements of the symbol set are ignored; however, if a cell library is not specified or the SYMBOL command gives a cell name which is not defined in the cell library, a SIF orphan cell is created and a message is sent to the message file. The orphan cell is defined in the user's design file using the simple elements in the SYMBOL command set. This design file can then be run through the SIF CPC



```

MID/ON
IDE/AS=4, ID=25, USER DATA ON A
IDE/AS=4, ID=25, COMPLEX STRING
MID/OF
BST/OP
LAC/LS=4
OVR/10
LST/OP, 100, 100, 200, 200, 300, 300
ARC/P1=300, 300, P2=350, 250, P3=300, 200
LST/300, 200, 400, 200, 500, 200
EST/

```

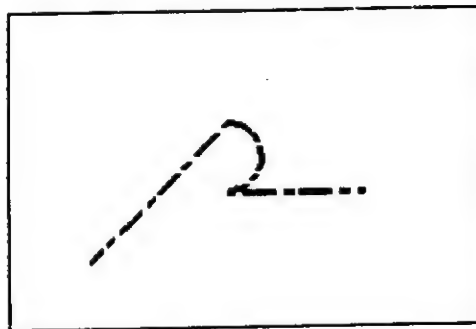


Figure 8-23. Example of SIF Complex String

The following example attempts to define a complex shape. Since the first vertex of the first internal element is not the same as the last vertex of the last internal element, the definition is invalid.

```

OVR/8
FNT/0
BST/SO
CNC/500, 50, 510, 60, 520, 70, 530, 80
CUR/530, 80, 540, 90, 550, 100, 600, 200, 700, 200
LAC/LS=3, LC=5
LST/OP, 700, 200, 1000, 50
EST/

```

#### B.11 END COMPLEX STRING Command - Generate Complex String or Complex Shape

The SIF END COMPLEX STRING command must be used to indicate the end of a complex string or complex shape definition. The END COMPLEX STRING command cannot be used without prior use of a SIF BEGIN COMPLEX STRING command. (See Section B.10.)

### 8.12.3 BEGIN SYMBOL Command - Interface Form

The interface form allows you to generate a BEGIN SYMBOL command by specifying certain options for the symbol set. The subroutine formats the command and writes it to a SIF output file. The following examples describe the calling sequence that is used when generating a SIF BEGIN SYMBOL command using the interface form:

```
CALL CMD52(stype)
```

```
CALL CMD52A(stype, nc, desc)
```

stype - Integer\*2 symbol set option  
0 - normal symbol  
1 - masked symbol  
2 - create symbol

nc - Integer\*2 number of characters in the description

desc - byte array containing cell description (used only when creating a cell definition)

### 8.12.4 BEGIN SYMBOL Command Restrictions

The restrictions of the BEGIN SYMBOL command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN SYMBOL command must always be used in conjunction with the END SYMBOL command. The SIF commands that can appear in a SYMBOL command set are the OVERLAY, CLASSIFICATION, FONT, LINE/AREA CHARACTERISTICS (line style, line weight, and line color only), TEMPORARY ORIGIN, TEXT LINE CHARACTERISTICS, PARAGRAPH CHARACTERISTICS, and GRAPHIC ELEMENT GENERATION commands.

The SIF GENERATE SYMBOL set may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)

processor and the orphan cells are defined in the cell library specified and replaced in the design file as a regular cell. There is also an option to use the simple elements to create a cell definition and place it in the specified cell library but no regular cell is placed in the current design file.

#### 8.12.1 BEGIN SYMBOL Command - ASCII Form

The ASCII form of the BEGIN SYMBOL command is indicated by a command type of BSY followed by a slash (/). Examples of the SIF BEGIN SYMBOL command are as follows:

```
BSY/
BSY/MK
BSY/CR
BSY/CR, THIS IS A CELL DESCRIPTION
```

The CR keyword indicates that you want to create a cell definition to be placed in the cell library specified in the environment file. The cell definition is created using the SYMBOL command information and the simple elements which follow in the symbol set. If text follows the CR keyword, the first 27 characters are placed in the cell definition as a cell description. If this keyword is specified, the cell definition is not completed and placed in the specified cell library until the SIF CPC processor is executed on the design file created after using one on the SIF-in processors (ATG or ASI/TRI). The cell is then defined in the cell library only, and is not found in the resulting design file. The CR keyword is optional.

The MK keyword indicates that the symbol definition which follows comes from a symbol which has been processed using the MASK user command. It has no effect on SIF-in processing and is an optional keyword.

#### 8.12.2 BEGIN SYMBOL Command - Binary Form

The binary form of the BEGIN SYMBOL command is indicated by a command type of 52 in the command header. The words to follow field indicates the number of Integer\*4 words in the description. The words to follow fields will either be 0 or 9 if a cell description is included. The subtype field indicates whether a cell with a description is to be created. The value field is always 0.

52	0
STYPE	0

```
stype = 0 - standard symbol placement
        1 - masked symbol
        2 - create cell definition and description and
           put in cell library
```

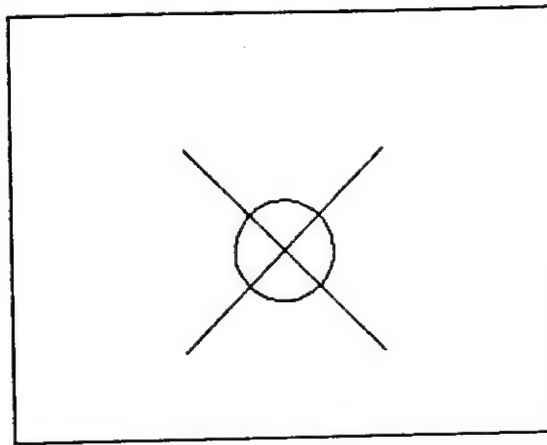


Figure 8-24. Example of SIF Expanded Symbol

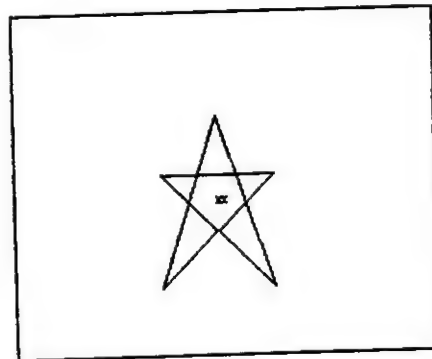


Figure 8-25. Example of SIF Created Symbol

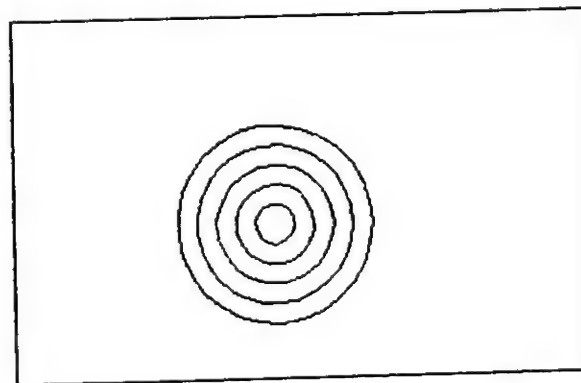


Figure 8-26. Example of SIF Created Symbol

### 8.12.5 BEGIN SYMBOL Command Examples

This section presents examples of the SIF BEGIN SYMBOL command. A description is given first followed by the example itself. All examples are given in the ASCII form.

The following example places a symbol named VANE with origin at 100,100. The components of the cell are two line strings and a circle. The components are ignored by SIF-in processors. The cell VANE is defined in the cell library specified in the environment file. Figure 8-24 is a graphic depiction of the following SIF ASCII commands:

```
BSY/  
SYM/OR=100,100 VANE  
LST/0,0,200,200  
LST/200,0,0,200  
CIR/CE=100,100,RA=50  
ESY/
```

The following example creates a cell named STAR to be placed in the cell library specified in the environment file. The cell is composed of two line strings and one text string. The cell is not placed in the design file, only in the cell library. The cell definition contains the description given. Figure 8-25 is a graphic depiction of the following SIF ASCII commands:

```
BSY/CR,STAR CELL  
SYM/OR=100,150,STAR  
LST/0,0,100,300,200,0  
LST/200,0,0,200,200,200,0,0  
TXT/OR=100,150,AN=0,TH=10,TW=10,XX  
ESY/
```

The following example attempts to create a cell definition of concentric circles. Figure 8-26 is a graphic representation of the following SIF ASCII commands:

```
BSY/CR,CONCENTRIC CIRCLES  
SYM/OR=1000,1000,CONCIR  
CIR/CE=0,0,RA=10  
CIR/CE=0,0,RA=20  
CIR/CE=0,0,RA=30  
CIR/CE=0,0,RA=40  
CIR/CE=0,0,RA=50  
ESY/
```

## B.14 BEGIN SURFACE Command - Generate Surface Element

The SIF SURFACE command set is supported in IGDS as a surface element type 18 and definition component elements. The surface sets are found in 3-D files only and consist of the BSF and ESF commands which specify the beginning and end of the surface element. Between the BSF and ESF commands are the graphic components which define the complex surface element. The components which make up the surface element must adhere to the guidelines as specified in the IGDS Application Software Interface Document. A surface type keyword is specified on the BSF command to identify the specific kind of surface being defined.

### B.14.1 BEGIN SURFACE Command - ASCII Form

The ASCII form of the BEGIN SURFACE command is indicated by a command type of BSF followed by a slash (/):

BSF/ST=stype

The ST keyword defines the surface type desired and is optional. If not specified, the ST keyword defaults to 0 (surface of projection).

### B.14.2 BEGIN SURFACE Command - Binary Form

The binary form of the BEGIN SURFACE command is indicated by a command type of 100 in the command header. The subtype field indicates the type surface represented and the value field is 0.

100	0
stype	0

### B.14.3 BEGIN SURFACE Command - Interface Form

The interface form of the BEGIN SURFACE command allows you to generate a BEGIN SURFACE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

If you do not specify a cell library in the environment file so SIF-in cannot create a cell definition; however, it does place an orphan cell in the design file which can be processed later using the CPC (create and place cell) processor.

### 8.13 END SYMBOL Command

The SIF END SYMBOL command must be used to indicate the end of a symbol set definition. The END SYMBOL command cannot be used without prior use of a SIF BEGIN SYMBOL command. (See Section 8.12.)

#### 8.13.1 END SYMBOL Command - ASCII Form

The ASCII form of the END SYMBOL command is indicated by a command type of ESY followed by a slash (/):

ESY/

#### 8.13.2 END SYMBOL Command - Binary Form

The binary form of the END SYMBOL command is indicated by a command type of 53 in the command header. All other fields are zero.

#### 8.13.3 END SYMBOL Command - Interface Form

The interface form allows you to generate an END SYMBOL command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD53

#### 8.13.4 END SYMBOL Command Restrictions

See Section 8.12.4.

#### 8.13.5 END SYMBOL Command Examples

See Section 8.12.5.

53	0
0	0

The following SIF ASCII commands generate a 3-D surface element. Figure 8-27 is a graphic representation of the following SIF ASCII commands:

```
BSF/ST=0
LST/OP, 1269224198, 1269214004, 1269739664, 1269224198, 1269150504,
1269739664
ARC/CC, CE=1269271823, 1269150504, 1269739650, P1=1269319448, 1269150504,
1269739650, P2=1269224198, 1269150504, 1269739650, MA=1., 0., 0., 0., 1., 0.,
0., 0., 1.
LST/OP, 1269319448, 1269150504, 1269739664, 1269319448, 1269214004,
1269739664
ARC/CC, CE=1269271823, 1269214004, 1269739650, P1=1269319448, 1269214004,
1269739650, P2=1269224198, 1269214004, 1269739650, MA=1., 0., 0., 0., 1., 0.,
0., 0., 1.
LST/OP, 1269224198, 1269214004, 1269712994, 1269224198, 1269150504,
1269712994
ARC/CC, CE=1269271823, 1269150504, 1269712980, P1=1269319448, 1269150504,
1269712980, P2=1269224198, 1269150504, 1269712980, MA=1., 0., 0., 0., 1., 0.,
0., 0., 1.
LST/OP, 1269319448, 1269150504, 1269712994, 1269319448, 1269214004,
1269712994
ARC/CC, CE=1269271823, 1269214004, 1269712979, P1=1269319448, 1269214004,
1269712979, P2=1269224198, 1269214004, 1269712979, MA=1., 0., 0., 0., 1., 0.,
0., 0., 1.
LST/OP, 1269224198, 1269214004, 1269739664, 1269224198, 1269214004,
1269712994
LST/OP, 1269224198, 1269150504, 1269739664, 1269224198, 1269150504,
1269712994
LST/OP, 1269224198, 1269150504, 1269739650, 1269224198, 1269150504,
1269712980
LST/OP, 1269266845, 1269103140, 1269739650, 1269266845, 1269103140,
1269712980
LST/OP, 1269319187, 1269145526, 1269739650, 1269319187, 1269145526,
1269712980
```



CALL CMD100(stype)

stype - Integer\*2 surface type  
0 - Surface of projection  
1 - Bounded plane  
2 - Unbounded plane  
3 - Right circular cylinder  
4 - Right circular cone  
5 - Tabulated cylinder  
6 - Tabulated cone  
7 - Convolute  
8 - Surface of revolution  
9 - Warped surface

#### 8.14.4 BEGIN SURFACE Command Restrictions

This section describes the restrictions of the BEGIN SURFACE command. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN SURFACE command is found in 3-D files only and must always be accompanied by an END SURFACE command (ESF). The BEGIN SURFACE command is found anywhere in the 3-D SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between a B-SPLINE CURVE or SURFACE command set (BBC/ and EBC/) or (BBS/ and EBS/)
- o between another BEGIN SURFACE (BSF) and an END SURFACE command (ESF)

#### 8.14.5 BEGIN SURFACE Command Examples

This section presents examples of the SIF BEGIN SURFACE command. An example of the 3-D SIF ASCII commands making up the surface set is given along with an IGDS graphic illustration of the element generated.

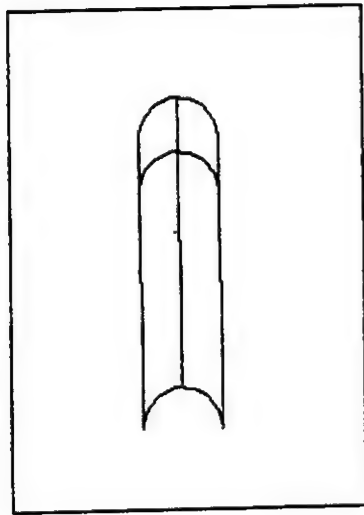


Figure 8-27. Example of SIF Surface

LST/OP, 1269319448, 1269150504, 1269739650, 1269319448, 1269150504,  
1269712980  
LST/OP, 1269319448, 1269150504, 1269739664, 1269319448, 1269150504,  
1269712994  
LST/OP, 1269319448, 1269214004, 1269739664, 1269319448, 1269214004,  
1269712994  
LST/OP, 1269319448, 1269214004, 1269739650, 1269319448, 1269214004,  
1269712980  
LST/OP, 1269271823, 1269261629, 1269739650, 1269271823, 1269261629,  
1269712980  
LST/OP, 1269224198, 1269214004, 1269739650, 1269224198, 1269214004,  
1269712980  
LST/OP, 1269224198, 1269214004, 1269739650, 1269224198, 1269214004,  
1269712980  
ESF/

## 8.16 BEGIN SOLID Command - Generate Capped Surface Element

The SIF SOLID command set is supported in IGDS as capped surface element type 19 and its definition component elements. The solid sets are found in 3-D files only and consist of the BSO and ESO commands which specify the beginning and end of the solid element. Between the BSO and ESO commands are the graphic components which define the capped surface element. The components which make up the capped surface must adhere to the guidelines as specified in the IGDS Application Software Interface Document. A solid type keyword is specified on the BSO command to identify the specific kind of capped surface being defined.

### 8.16.1 BEGIN SOLID Command - ASCII Form

The ASCII form of the BEGIN SOLID command is indicated by a command type of BSO followed by a slash (/):

BSO/ST=stype

The ST keyword defines the solid type desired and is optional. If not specified, the ST keyword defaults to 0 (volume of projection).

### 8.16.2 BEGIN SOLID Command - Binary Form

The binary form of the BEGIN SOLID command is indicated by a command type of 102 in the command header. The subtype field indicates the type of capped surface represented and the value field is 0.

102	0
stype	0

### 8.16.3 BEGIN SOLID Command - Interface Form

The interface form allows you to generate a BEGIN SOLID command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD102 (stype)

stype - Integer\*2 solid type

0 - Volume of projection

1 - Volume of revolution

2 - Volume defined by boundary elements

## 8.15 END SURFACE Command

The SIF END SURFACE command must be used to indicate the end of a surface set definition. The END SURFACE command may not be used without prior use of a SIF BEGIN SURFACE command. (See Section 8.14.) The SIF ASCII, binary, and interface forms of the command follow.

### 8.15.1 END SURFACE Command - ASCII Form

The ASCII form of the END SURFACE command is indicated by a command type of ESF followed by a slash (/):

ESF/

### 8.15.2 END SURFACE Command - Binary Form

The binary form of the END SURFACE command is indicated by a command type of 101 in the command header. All other fields are zero.

101	0
0	0

### 8.15.3 END SURFACE Command - Interface Form

The interface form allows you to generate an END SURFACE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD101

### 8.15.4 END SURFACE Command Restrictions

See Section 8.14.4.

### 8.15.5 END SURFACE Command Examples

See Section 8.14.5.

LST/OP, 1269891433, 1269261502, 1269936500, 1269891433, 1269261502,  
1269909830  
LST/OP, 1269843173, 1269309762, 1269936500, 1269843173, 1269309762,  
1269909830  
LST/OP, 1269794913, 1269261502, 1269936500, 1269794913, 1269261502,  
1269909830  
LST/OP, 1269843173, 1269213242, 1269936500, 1269843173, 1269213242,  
1269909830  
ESO/

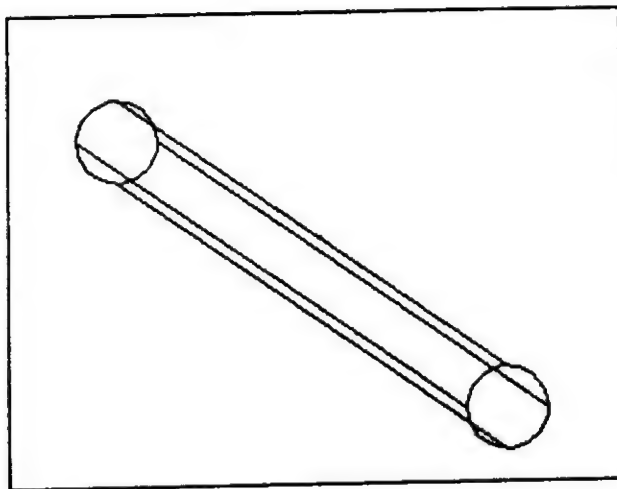


Figure 8-28. Example of SIF Solid

#### B.16.4 BEGIN SOLID Command Restrictions

This section describes the restrictions of the BEGIN SOLID command. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN SOLID command is found in 3-D files only and must always be accompanied by an END SOLID command (ESO) that follows the graphic commands. The BEGIN SOLID command is found anywhere in the 3-D SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING commands (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before and INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between a B-SPLINE CURVE or B-SPLINE SURFACE command set (BBC/ and EBC/) or (BBS/and EBS/)
- o between another BEGIN SOLID (BSO/) and END SOLID (ESO/) command

#### B.16.5 BEGIN SURFACE Command Examples

This section presents examples of the SIF BEGIN SOLID command. An example of the 3-D SIF ASCII commands making up the solid set is given along with an IGDS graphic illustration of the element generated.

Figure B-28 is a graphic representation of the following SIF ASCII commands which generate a 3-D SIF solid element:

```
BSO/ST=0
CIR/CE=1269843173,1269261502,1269936500,P1=1269891432,1269261502,
1269936500,P2=1269843173,1269309761,1269936500,MA=1.,0.,0.,0.,1.,0.,
0.,0.,1.
CIR/CE=1269843173,1269261502,1269909830,P1=1269891432,1269261502,
1269909830,P2=1269843173,1269309761,1269909830,MA=1.,0.,0.,0.,1.,0.,
0.,0.,1.
```

## 8.18 POINT STRING Command - Generate Point String

The POINT STRING (PST) command can be used to create an IGDS point string element type 22. Point strings may be continuous or disjointed. The points of a continuous point string are considered to be joined together in the order the points are defined in the command, while the points of a disjointed point string are not considered to be joined together. The orientation of the points for both 2-D and 3-D point strings is a transformation matrix. The ASCII, binary, and interface forms of the POINT STRING command are presented in the following sections. All coordinates must be entered in units of resolution (UDRs). The maximum number of points in a POINT STRING command is 48.

### 8.18.1 POINT STRING Command - ASCII Form

The ASCII form of the POINT STRING command is indicated by a command type of PST followed by a slash (/):

```
PST/DJ,OR,x1,y1,m1,x2,y2,m2,...,xn,yn,mn
PST/CO,NO,x1,y1,x2,y2,...,xn,yn
PST/DJ,OR,x1,y1,z1,m1,x2,y2,z2,m2,...,xn,yn,zn,mn
PST/CO,NO,x1,y1,z1,x2,y2,z2,...,xn,yn,zn
```

The CO and DJ keywords indicate whether the point string is continuous (CO) or disjointed (DJ) with the default being continuous. The NO and OR keywords indicate whether there are orientations specified (OR) or not (NO) with the default being that orientations are specified. The "m1", "m2", and "mn" are the orientation matrices for each point. If orientations are not specified, the identity matrix is used.



### 8.17 END SOLID Command

The SIF END SOLID command must be used to indicate the end of a SOLID set definition. The END SOLID command may not be used without prior use of a SIF BEGIN SOLID command. (See Section 8.16.) The SIF ASCII, binary and interface forms of the command follow.

#### 8.17.1 END SOLID Command - ASCII Form

The ASCII form of the END SOLID command is indicated by a command type of ESO followed by a slash (/):

ESO/

#### 8.17.2 END SOLID Command - Binary Form

The binary form of the END SOLID command is indicated by a command type of 103 in the command header. All other fields are zero.

103	0
0	0

#### 8.17.3 END SOLID Command - Interface Form

The interface form allows you to generate an END SOLID command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD103

#### 8.17.4 END SOLID Command Restrictions

See Section 8.16.4.

#### 8.17.5 END SOLID Command Examples

See Section 8.16.5.

### 8.18.3 POINT STRING Command - Interface Form

The interface form of the POINT STRING command allows you to generate a POINT STRING command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

```
CALL CMD46(stype,value,nv,vert,orient)
```

stype - Integer\*2 type of point string.  
1 - continuous  
2 - disjointed

value - Integer\*2 orientation indicator.  
0 - no orientations are specified  
1 - orientations are specified

nv - Integer\*2 number of vertices in point string. If "nv" is greater than 0, the "vert" and "orient" arrays contain all vertices for the point string. If "nv" is equal to 0, an unknown number of vertices and orientations are to be sent in one or more calls to SIFPTS. If "nv" is less than 0, a known number of vertices and orientations are to be sent in one or more calls to SIFPTS.

vert - Integer\*4 array containing the coordinates of the point string. Two coordinates are required in a 2-D point string, and three coordinates are required in a 3-D point string.

orient- Real\*4 array defining the orientation matrix for each vertex in the point string. This matrix is required for each point if value is set to 1. Four terms define the matrix for a 2-D point string, and nine terms define the matrix for a 3-D point string.

### 8.18.4 POINT STRING Command Restrictions

The restrictions of the POINT STRING command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The SIF POINT STRING command is limited to 48 points.

The SIF POINT STRING command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)

### 8.18.2 POINT STRING Command - Binary Form

The binary form of the POINT STRING command is indicated by a command type of 46 in the command header. The subtype field indicates whether the point string is continuous or disjoint and the values field indicates the presence of orientations.

46	#words
stype	value
x1	
y1	
t11 *10000.	
t21 *10000.	
t12 *10000.	
t22 *10000.	

stype - type of point string.

- 1 - continuous
- 2 - disjointed

value - orientation indicator

- 0 - no orientations are specified
- 1 - orientations are specified

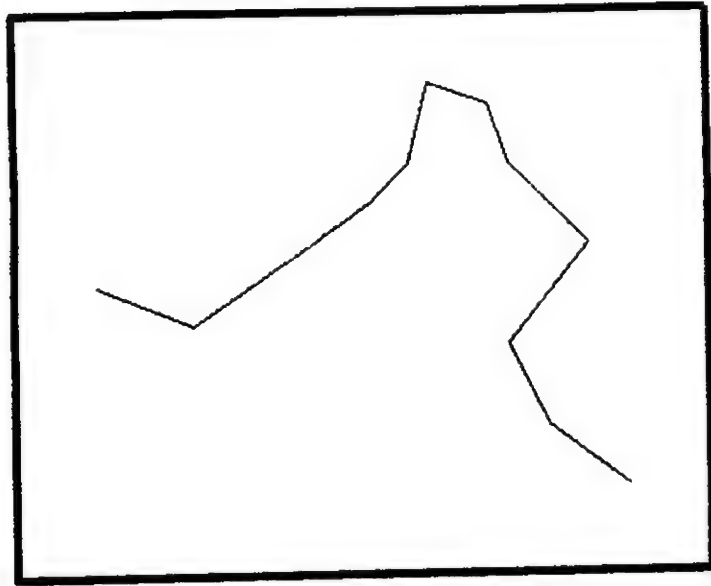


Figure 8-29. Example of SIF 2-D Point String

- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a B-SPLINE CURVE or B-SPLINE SURFACE command set (BBC/, EBC/ or BBS/, EBS/)

#### B.18.5 POINT STRING Command Examples

This section presents examples of the SIF POINT STRING command. Both 2-D and 3-D POINT STRING examples are given in their SIF ASCII form along with an IGDS graphic illustration of the elements generated.

The following SIF ASCII commands generate a 2-D point string that is continuous. The point string is placed with an orientation using the identity matrix. Figure B-29 is a graphic representation of the following SIF ASCII command:

```
PST/CO,OR,100,1200,1.,0.,0.,1.,600,1000,1.,0.,0.,1.,
900,1200,1.,0.,0.,1.,1200,1400,1.,0.,0.,1.,
1500,1600,1.,0.,0.,1.,1700,1800,1.,0.,0.,1.,
1800,2200,1.,0.,0.,1.,2100,2100,1.,0.,0.,1.,
2200,1800,1.,0.,0.,1.,2600,1400,1.,0.,0.,1.,
2200,900,1.,0.,0.,1.,2400,500,1.,0.,0.,1.,
2800,200,1.,0.,0.,1.
```

The following SIF ASCII commands place a 3-D point string as continuous and with an orientation matrix. Figure B-30 is a graphic representation of the following SIF ASCII command:

```
PST/CO,OR,0,0,0.,707,0.,0.,0.,1.,0.,0.,0.,707,
400,300,100.,707,0.,0.,0.,1.,0.,0.,0.,707,
700,1000,200.,707,0.,0.,0.,1.,0.,0.,0.,707,
900,1200,2100.,707,0.,0.,0.,1.,0.,0.,0.,707,
1000,1300,400.,707,0.,0.,0.,1.,0.,0.,0.,707,
800,1700,4200.,707,0.,0.,0.,1.,0.,0.,0.,707,
200,1600,400.,707,0.,0.,0.,1.,0.,0.,0.,707,
100,1100,100.,707,0.,0.,0.,1.,0.,0.,0.,707,
500,900,100.,707,0.,0.,0.,1.,0.,0.,0.,707,
1100,300,100.,707,0.,0.,0.,1.,0.,0.,0.,707
```

## B.19 CYLINDER Command - Generate Circular Culinder

The CYLINDER (CYL) command is supported in IGDS as a circular truncated cone element (type 23). The CYLINDER command defines the center point of the base of the cylinder, the radius of the base, the center point of the top of the cylinder, the specific type of cylinder, and a transformation matrix. The matrix applies to both the top and base of the cylinder to ensure that they always lie in parallel planes. The CYLINDER command is supported only in 3-D SIF files. The cylinder type may be specified with one of the following codes:

- o right cylinder (the centerline forms right angles with the planes of the base and top circles.
- o cylinder (the centerline forms an angle other than 90 degrees with the planes of the top and base circles.

### B.19.1 CYLINDER Command - ASCII Form

The ASCII form of the CYLINDER command is indicated by a command type of CYL followed by a slash (/):

```
CYL/SO,CT=ctype,BC=bocen,BR=borad,TC=tocen,MA=matrix  
CYL/SF,CT=ctype,BC=bocen,BR=borad,TC=tocen,MA=matrix
```

The SO and SF keywords define the cylinder type and are optional. The SO keyword indicates that the cylinder is a solid and is the default if neither keyword is specified. The SF keyword indicates that the cylinder is a surface. The CT keyword defines the type of cylinder by certain characteristics and defaults to 1 if not specified. The BC keyword defines the center of the circle describing the base of the cylinder in UORs and is required. The BR keyword defines the radius of the circle describing the base of the cylinder in UORs and is required. The TC keyword defines the center of the circle describing the top of the cylinder in UORs and is required. The MA keyword defines the transformation matrix to be used on both the base and top circles and is optional. The MA keyword may be omitted if no matrix is desired or if the MTX command appears immediately before the CYLINDER command.

### B.19.2 CYLINDER Command - Binary Form

The binary form of the CYLINDER command is indicated by a command type of 56 in the command header. The subtype field indicates the type cylinder represented and the values field indicates whether the cylinder is a surface or a solid.

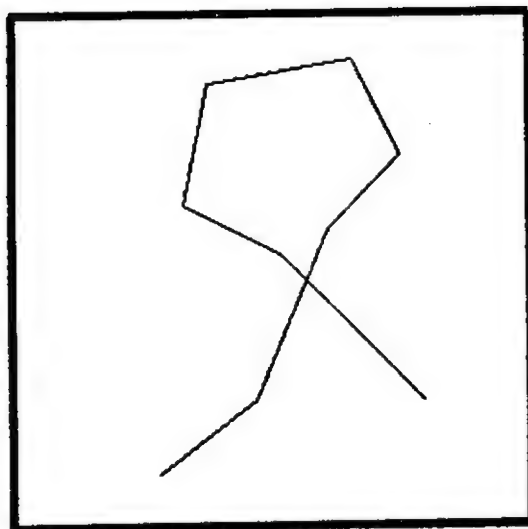


Figure 8-30. Example of SIF 3-D Point String

t12
*10000.
t22
*10000.
t32
*10000.
t13
*10000.
t23
*10000.
t33
*10000.



56	#words
stype	value
x bottom	
y bottom	
z bottom	
bottom radius	
x top	
y top	
z top	
t11 *10000.	
t21 *10000.	
t31 *10000.	

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between a B-SPLINE CURVE or B-SPLINE SURFACE command set (BBC/, EBC/ or BBS/, EBS/)

#### B.19.5 CYLINDER Command Examples

This section presents examples of the SIF CYLINDER command. An example of the 3-D SIF ASCII command is given along with an IGDS graphic illustration of the element generated.

The following SIF ASCII command generates a 3-D cylinder as a solid. Figure 8-31 is a graphic illustration of the following SIF ASCII command:

```
CYL/SD,CT=1,BC=600,600,600,BR=100,TC=1400,1400,1400,MA=1.,
0.,0.,0.,707,0.,0.,0.,707
```

stype - cylinder type (1-2). A bias of 100 is added to this value if a matrix is present.

value - solid or surface indicator.  
0 - solid  
1 - surface

### B.19.3 CYLINDER Command - Interface Form

The interface form allows you to generate a CLYINDER command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

```
CALL CMD56(stype,value,bocen,borad,tocen,mflg,matrix)
```

stype - Integer\*2 cylinder type.

1 - right cylinder  
2 - cylinder

value - Integer\*2 solid or surface indicator.  
0 - solid  
1 - surface

bocen - Integer\*4 array defining the center point of the base circle in UORs.

borad - Integer\*4 value defining the radius of the base circle in UORs.

tocen - Integer\*4 array defining the center point of the top circle in UORs.

mflg - Integer\*2 matrix indicator.  
0 - no matrix is present  
1 - matrix present

matrix- Real\*4 array containing the nine terms of the transformation matrix.

### B.19.4 CYLINDER Command Restrictions

The restrictions of the CYLINDER command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The cylinder may appear in 3-D SIF files only. The CYLINDER command may appear anywhere in the 3-D SIF file except for the following conditions:

The following SIF ASCII command generates a 3-D cylinder as a surface. Figure 8-32 is a graphic illustration of the following SIF ASCII command:

```
CYL/SF, CT=2, BC=600, 600, 1500, BR=1400, TC=1000, 1300, 100, MA=.707,  
0., 0., 0., .707, 0., 0., 0., 1.
```

#### **8.20 CIRCULAR TRUNCATED CONE Command - Generate Circular Truncated Cone**

The CIRCULAR TRUNCATED CONE (CTC) command is supported in IGDS as a circular truncated cone element (type 23). The CIRCULAR CONE command defines the center point of the base of the cone, the radius of the base, the center point of the top of the cone, the radius of the top, the specific type of cone and a transformation matrix. The matrix applies to both the top and the base of the cone to ensure that they always lie in parallel planes. The CIRCULAR CONE command is supported only in 3-D SIF files. The cone type may be specified with one of the following codes:

- 0 - general non-specific cone (not pointed or truncated).
- 3 - right cone (One radius of the cone is equal to zero, and the centerline is at a right angle to the planes of the base and the top.)
- 4 - cone (One radius of the cone is equal to zero and the centerline forms an angle other than 90 degrees with the planes of the base and the top.)
- 5 - right truncated cone (The radii of the cone are unequal and non-zero, and the centerline forms a right angle with the planes of the base and the top.)
- 6 - truncated cone (The radii of the cone are unequal and non-zero, and the centerline forms an angle other than 90 degrees with the planes of the base and the top.)

##### **8.20.1 CIRCULAR TRUNCATED CONE Command - ASCII Form**

The ASCII form of the CIRCULAR TRUNCATED CONE command is indicated by a command type of CTC followed by a slash (/):

```
CTC/SO, CT=ctype, BC=bocen, BR=borad, TC=tocen, TR=torad,  
MA=matrix  
CTC/SF, CT=ctype, BC=bocen, BR=borad, TC=tocen, TR=torad,  
MA=matrix
```

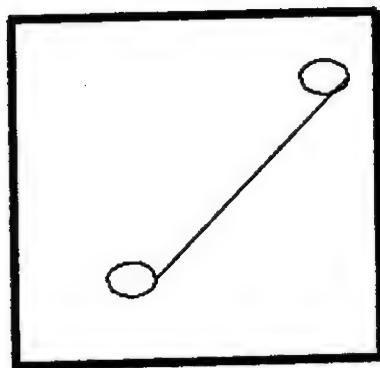


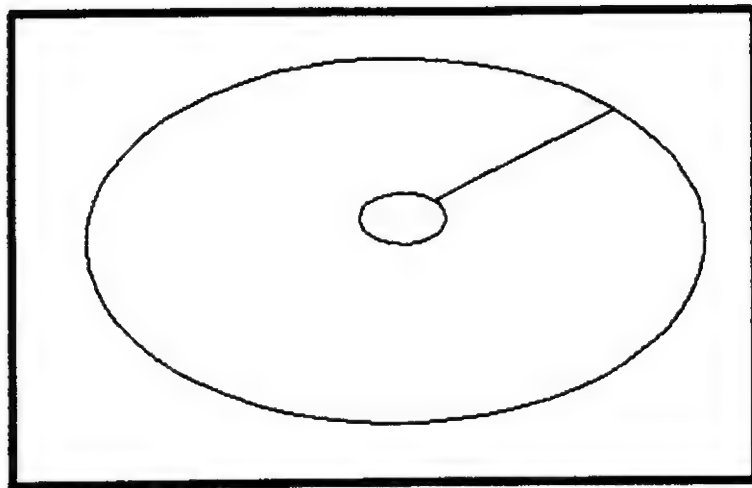
Figure 8-31. Example of 3-D SIF Cylinder as a Solid

The SO and SF keywords define the cone type and are optional. The SO keyword indicates that the cone is a solid and is the default if neither keyword is specified. The SF keyword indicates that the cone is a surface. The CT keyword defines the type of cone by certain characteristics and defaults to 0 if not specified. The BC keyword defines the center of the circle describing the base of the cone in UORs and is required. The BR keyword defines the radius of the circle describing the base of the cone in UORs and is required. The TC keyword defines the center of the circle describing the top of the cone in UORs and is required. The TR keyword defines the radius of the circle describing the top of the cone in UORs and is required. The MA keyword defines the transformation matrix to be used on both the base and the top circles and is optional. The MA keyword may be omitted if no matrix is desired and if the MTX command appears immediately before the CONE command.

#### B.20.2 CIRCULAR TRUNCATED CONE Command - Binary Form

The binary form of the CIRCULAR TRUNCATED CONE command is indicated by a command type of 55 in the command header. The subtype field indicates the type cone represented and the values field indicates whether the cone is a solid or a surface.

55	#words
stype	value
x bottom	
y bottom	
z bottom	
bottom radius	
x top	
y top	



**Figure 8-32. Example of 3-D SIF Cylinder as a Solid Surface**

### 8.20.3 CIRCULAR TRUNCATED CONE Command - Interface Form

The interface form allows you to generate a CIRCULAR TRUNCATED CONE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

```
CALL CMD55(stype,value,bocen,borad,tocen,torad,mflg,  
           matrix)
```

stype - Integer\*2 cone type (0,3-6).

value - Integer\*2 solid or surface indicator.  
0 - solid  
1 - surface

bocen - Integer\*4 array defining the center point of the base circle in UORs.

borad - Integer\*4 value defining the radius of the base circle in UORs.

tocen - Integer\*4 array defining the center point of the top circle in UORs.

torad - Integer\*4 value defining the radius of the top circle in UORs.

mflg - Integer\*2 matrix indicator.  
0 - no matrix is present  
1 - matrix present

matrix - Real\*4 array containing the nine terms of the transformation matrix.

### 8.20.4 CIRCULAR TRUNCATED CONE Command Restrictions

The restrictions of the CIRCULAR TRUNCATED CONE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The CIRCULAR TRUNCATED CONE command may appear in 3-D SIF files only. The command may appear anywhere in the 3-D SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)



z top
top radius
t11 *10000.
t21 *10000.
t31 *10000.
t12 *10000.
t22 *10000.
t32 *10000.
t13 *10000.
t23 *10000.
t33 *10000.

stype - cone (0,3-6). A bias of 100 is added to this value  
is a matrix is present.

value - solid or surface indicator.  
0 - solid  
1 - surface

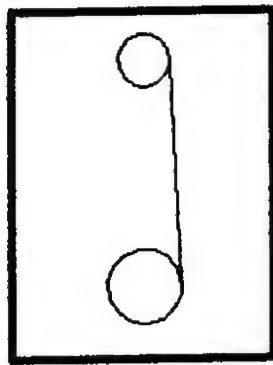


Figure 8-33. Example of 3-D SIF Cone as a Solid

- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between a B-SPLINE CURVE or B-SPLINE SURFACE command set (BBC/, EBC/ or BBS/, EBS/)

#### B.20.5 CIRCULAR TRUNCATED CONE Command Examples

This section presents examples of the SIF CIRCULAR TRUNCATED CONE command. An example of the 3-D SIF ASCII command is given along with an IGDS graphic illustration of the element generated.

The following SIF ASCII command generates a 3-D circular truncated cone as a solid, general, non-specific cone. Figure B-33 is a graphic representation of the following SIF ASCII command:

```
CTC/SD, CT=0, BC=100, 100, 0, BR=50, TC=100, 400, 0, TR=35, MA=1., 0., 0.,
0., 1., 0., 0., 0., 1
```

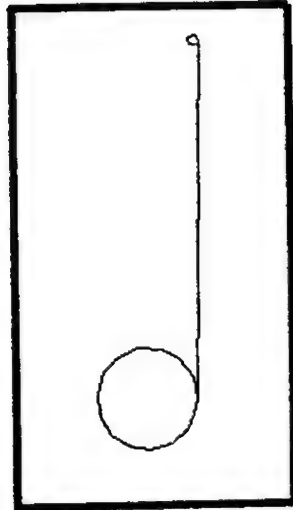


Figure 8-34. Example of 3-D SIF Cone as a Surface

The B-spline curve element may be open, closed, or closed representing a hole. You may specify the order of the B-spline curve from 2 to 16. You may also specify the B-spline curve type according to the following codes:

- 0 - general B-spline curve
- 1 - line
- 2 - circular arc
- 3 - circle
- 4 - elliptical arc
- 5 - ellipse
- 6 - parabolic arc
- 7 - hyperbolic arc

The following SIF ASCII command places a circular truncated cone as a surface representing a right cone. Figure B-34 is a graphic representation of the following SIF ASCII command:

```
CTC/SF, CT=3, BC=500, 500, 1000, BR=500, TC=1000, 4000, 1000, TR=50,
MA=1., 0., 0., 0., 1., 0., 0., 0., 1.
```

## B.21 BEGIN B-SPLINE CURVE Command

The B-spline curve (BBC) is represented in SIF as a B-SPLINE CURVE command set and is recognized in both 2-D and 3-D SIF files. The format of the B-SPLINE CURVE command set requires a specific sequence of commands. No other command in the command set may appear without prior use of a BBC command. If the B-spline curve is non-uniform, the KNOT (KNO) command must immediately follow the BBC command. If the B-spline curve is uniform, the KNO command must not be used. The next command must be a POLE (POL) command to define the coordinates of the control polygon. The POL command is required for all B-spline curves. If the B-spline curve is rational, the WEIGHT (WEI) command must immediately follow the POL command. If the B-spline curve is non-rational, the WEI command must not be used. The next command must be an END B-SPLINE CURVE (EBC) command to indicate the end of the command set. The EBC command is required for all B-spline curves. These commands are described in more detail in the following sections. The recognized B-SPLINE CURVE commands sets are:

```
BBC/...
POL/...      (uniform, non-rational)
EBC/

BBC/...
KNO/...      (non-uniform, non-rational)
POL/...
EBC/

BBC/...
POL/...      (uniform, rational)
WEI/...
EBC/

BBC/...
KNO/...
POL/...      (non-uniform, rational)
WEI/...
EBC/
```

### B.21.3 BEGIN B-SPLINE CURVE Command - Interface Form

The interface form allows you to generate a BEGIN B-SPLINE CURVE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD120(stype, type, order, numk)

stype - Integer\*2 flag indicating B-spline curve status.

1 - open

2 - closed

3 - closed representing a hole

type - Integer\*2 element type (0-7).

order - Integer\*2 order of the B-spline curve (2-16).

numk - Integer\*2 number of knots for B-spline curve.

### B.21.4 BEGIN B-SPLINE CURVE Command Restrictions

The restrictions of the B-SPLINE CURVE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN B-SPLINE CURVE command must always be used in conjunction with the END B-SPLINE CURVE command. The only SIF commands that may appear in a B-SPLINE CURVE set are the KNOT command, POLE command, and WEIGHT command. The POLE command must appear within the set.

The SIF B-SPLINE CURVE command set may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between another B-SPLINE CURVE or B-SPLINE SURFACE command set (BBC/, EBC/ or BBS/, EBS/)

### B.21.1 BEGIN B-SPLINE CURVE Command - ASCII Form

The ASCII form of the BEGIN B-SPLINE CURVE command is indicated by a command type of BBC follows by a slash (/).

```
BBC/OP,ET=type,OR=order,KN=numk  
BBC/SO,ET=type,OR=order,KN=numk  
BBC/HO,ET=type,OR=order,KN=numk
```

The OP, SO, and HO keywords indicate whether the B-spline curve is open, closed, or closed representing a hole, and defaults to open if not specified. The ET keyword defines the curve type and defaults to 0 if not specified. The OR keyword defines the order of the B-spline and defaults to 2 if not specified. The KN keyword defines the number of knots if the B-spline curve is non-uniform and defaults to 0 for a uniform B-spline curve if not specified.

### B.21.2 BEGIN B-SPLINE CURVE Command - Binary Form

The binary form of the BEGIN B-SPLINE CURVE command is indicated by a command type of 120 in the command header. The subtype field indicates the status of the B-spline curve and the values field is 0.

stype - flag indicating B-spline curve status.  
1 - open  
2 - closed  
3 - closed representing a hole

120	3
stype	0
type	
order	
numk	

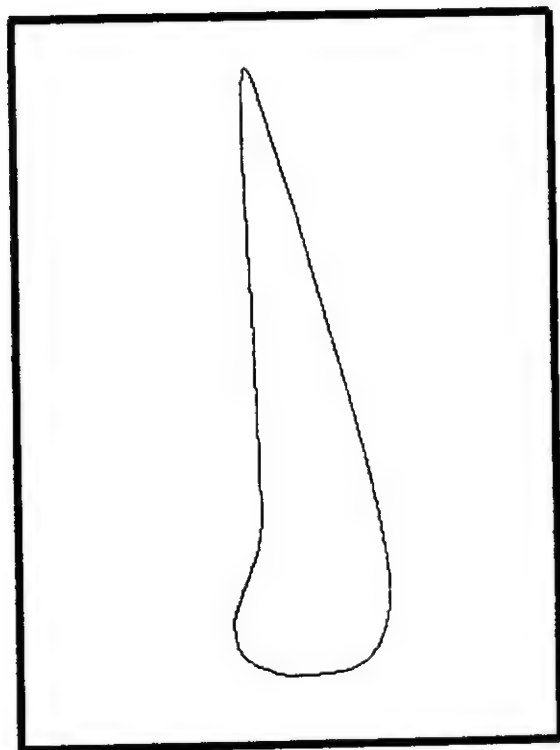


Figure 8-35. Example of 2-D B-Spline Curve  
(uniform, rational)



#### B.21.5 BEGIN B-SPLINE CURVE Command Examples

This section presents examples of the SIF B-SPLINE CURVE command. Both 2-D and 3-D B-spline curve examples are given in their SIF ASCII form along with an IGDS graphic representation of the elements generated.

The following example places a 2-D B-Spline curve as a hole which is uniform and rational representing a parabolic arc. Figure B-35 is a graphic illustration of the following SIF ASCII commands:

```
BBC/HO,ET=6,OR=10,KN=0
POL/100,200,3000,400,500,600,7000,800,900,1000,1100,12000,
    1300,1400,1500,1600,1700,1800,19000,2000
WEI/.5,.7,.2,.1,.2,.3,.4,.5,.1,.5
EBC/
```

The following example places a 2-D B-spline curve as uniform and non-rational representing an ellipse. Figure B-36 is a graphic illustration of the following SIF ASCII commands:

```
BBC/SO,ET=5
POL/100,100,200,200,300,100,200,0,100,100
EBC/
```

The following example places a 2-D B-spline curve as non-uniform and non-rational representing a general B-spline curve. Figure 8-37 is a graphic illustration of the following SIF ASCII commands:

```
BBC/OP,ET=0,OR=2,KN=4
KNO/.5,.5,.4,.5
POL/1000,1000,2000,1500,3000,1250,3500,1750,2500,1500,
    2700,1800
EBC/
```

The following example places a 2-D B-spline curve as non-uniform and rational representing a line. Figure 8-38 is a graphic illustration of the following SIF ASCII commands.

```
BBC/OP,ET=1,OR=2,KN=4
KNO/.6,.7,.6,.5
POL/1000,1000,2000,1500,3000,1250,3500,1750,2500,1500,
    2700,1800
WEI/.1,.1,.1,.1,.2,.1
EBC/
```

The following example places a 3-D B-spline curve as non-uniform and rational representing a parabolic arc. Figure 8-39 is a graphic representation of the following SIF ASCII commands:

```
BBC/OP,ET=6,OR=6,KN=2
KNO/.4,.3
POL/100,200,300,400,600,600,900,1000,1100,600,300,1400,
    200,100,1700,-200,50,2000,-400,0,1000,-800,-100,200
WEI/.5,.7,.2,.1,.2,.3,.4,.5
EBC/
```

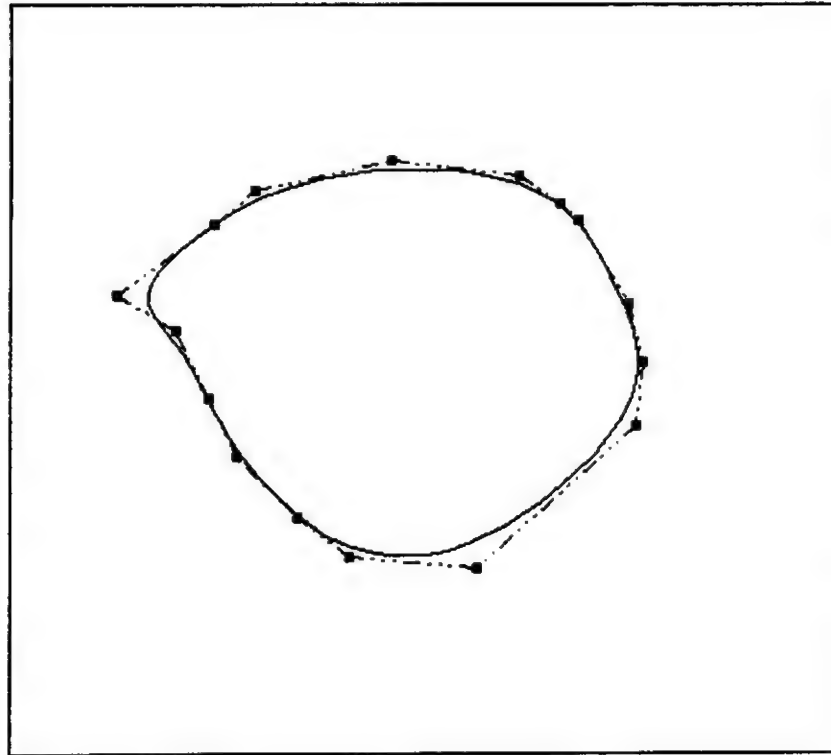


Figure 8-36. Example of 2-D SIF B-Spline Curve  
(uniform, non-rational)

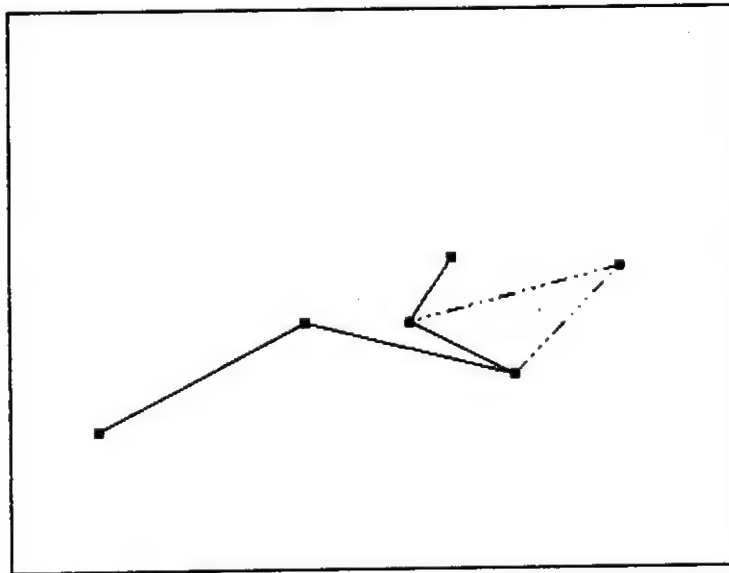


Figure 8-38. Example of 2-D B-Spline Curve  
(non-uniform, rational)

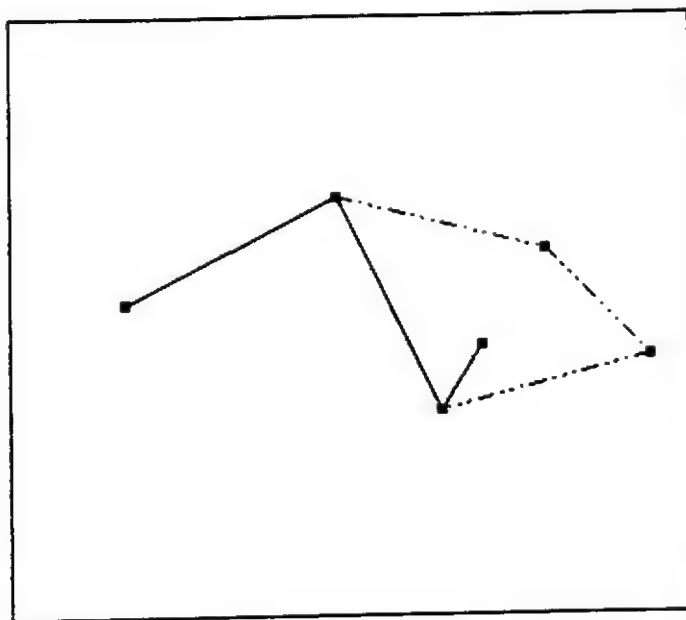


Figure B-37. Example of 2-D B-Spline Curve  
(non-uniform, non-rational)

## B.22 END B-SPLINE CURVE Command

The END B-SPLINE CURVE (EBC) command must be used to indicate the end of the B-spline curve definition. The EBC command may not be used without prior use of a BEGIN B-SPLINE CURVE (BBC) command.

### B.22.1 END B-SPLINE CURVE Command - ASCII Form

The ASCII form of the END B-SPLINE CURVE command is indicated by a command type of EBC followed by a slash (/):

EBC/

### B.22.2 END B-SPLINE CURVE Command - Binary Form

The binary form of the END B-SPLINE CURVE command is indicated by a command type of 121 in the command header. All other fields are 0.

121	0
0	0

### B.22.3 END B-SPLINE CURVE Command - Interface Form

The interface form allows you to generate an END B-SPLINE CURVE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD121

### B.22.4 END B-SPLINE CURVE Command Restrictions

See Section B.21.4.

### B.22.5 END B-SPLINE CURVE Command Examples

See Section B.21.5.

## B.23 BEGIN B-SPLINE SURFACE Command

The B-spline surface (BBS) is represented in SIF as a B-SPLINE SURFACE command set and is recognized only in 3-D SIF files. The format of the B-SPLINE SURFACE command set requires a specific sequence of commands. No other command in the command set may appear without prior use of a BBS command. If the B-spline surface is non-uniform, the KNOT (KNO) command

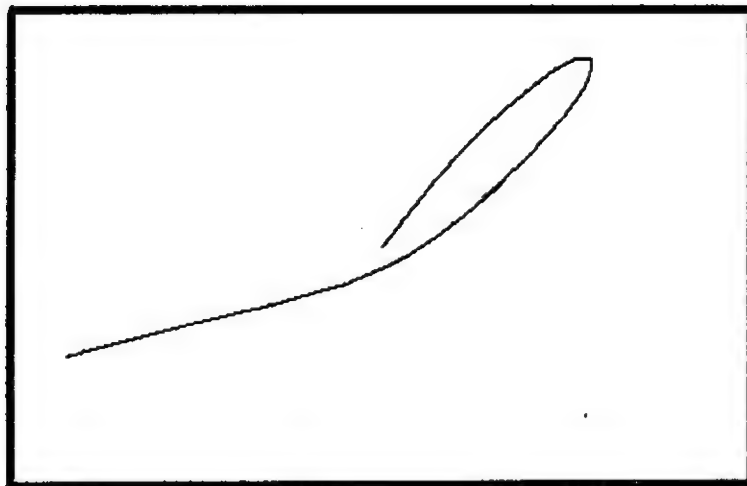


Figure B-39. Example of 3-D B-Spline Curve  
(non-uniform, rational)

- 4 - sphere
- 5 - torus
- 6 - surface of revolution
- 7 - tabulated cylinder
- 8 - ruled surface

#### 8.23.1 BEGIN B-SPLINE SURFACE Command - ASCII Form

The ASCII form of the BEGIN B-SPLINE SURFACE command is indicated by a command type of BBS followed by a slash (/):

```

BBS/OU, OV, ET=type, UD=uorder, VD=vorder, UR=urule, VR=vrule,
    UK=numuk, VK=numvk
BBS/SU, SV, ET=type, UD=uorder, VD=vorder, UR=urule, VR=vrule,
    UK=numuk, VK=numvk
BBS/HU, HV, ET=type, UD=uorder, VD=vorder, UR=urule, VR=vrule,
    UK=numuk, VK=numvk

```

The OU, SU, and HU keywords indicate the status of the B-spline surface in the U direction. OU indicates that the surface is open in the U direction and is the default if one of the keywords is not specified. SU indicates that the surface is closed in the U direction, and HU indicates that the surface is closed in the U direction and represents a hole. The OV, SV, and HV keywords indicate the status of the B-spline surface in the V direction. OV indicates that the surface is open in the V direction and is the default if one of the keywords is not specified. SV indicates that the surface is closed in the V direction, and HV indicates that the surface is closed in the V direction and represents a hole. The ET keyword defines the surface type and defaults to 0 if not specified. The UD keyword defines the order of the B-spline surface in the U direction and defaults to 2 if not specified. The VD keyword defines the order of the B-spline surface in the V direction and defaults to 2 if not specified. The UR keyword defines the number of rule lines in the U direction and defaults to 5 if not specified. The VR keyword defines the number of rule lines in the V direction and defaults to 5 if not specified. The UK keyword defines the number of knots in the U direction and defaults to 0 if not specified. The VK keyword defines the number of knots in the V direction and defaults to 0 if not specified.



must immediately follow the BBS command. If the B-spline surface is uniform, the KNOT command must not be used. If surface boundaries are specified, one or more BOUNDARY (BOU) commands must immediately follow the KNOT command (if knots are specified) or the BBS command. The next command must be a POLE (POL) command to define the coordinates of the control polygon. The POL command is required for all B-spline surfaces. If the B-spline surface is rational, the WEIGHT (WEI) command must immediately follow the POLE command. If the B-spline surface is non-rational, the WEIGHT command must not be used. The next command must be an END B-SPLINE SURFACE (EBS) command to indicate the end of the command set. The EBS command is required for all B-spline surfaces. These commands are described in more detail in the following sections. The recognized B-spline surface command sets follow. The BOU command is shown in each command set but is optional.

```

BBS/...
BOU/...      (uniform, non-rational)
POL/...
EBS/

BBS/...
KNO/...
BOU/...      (non-uniform, non-rational)
POL/...
EBS/

BBS/...
BOU/...
POL/...      (uniform, rational)
WEI/...
EBS/

BBS/...
KNO/...
BOU/...      (non-uniform, rational)
POL/...
WEI/...
EBS/

```

The B-spline surface element may be open, closed, or closed representing a hole. You can specify the order of the B-spline curve from 2 to 16. You can also specify the B-spline surface type according to the following codes:

- 0 - general B-spline surface
- 1 - plane
- 2 - right circular cylinder
- 3 - cone

### 8.23.3 BEGIN B-SPLINE SURFACE Command - Interface Form

The interface form of the BEGIN B-SPLINE SURFACE command you to generate a BEGIN B-SPLINE SURFACE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

```
CALL CMD122(stype,value,type,uorder,vorder,urule,vrule,
            numuk,numvk)
```

stype - flag indicating status in U direction.

- 1 - open
- 2 - closed
- 3 - closed representing a hole

value - flag indicating status in V direction.

- 1 - open
- 2 - closed
- 3 - closed representing a hole

type - element type (0-8).

uorder- order in U direction

vorder- order in V direction

urule - number of ruled lines in the U direction

vrule - number of ruled lines in the V direction

numuk - number of knots in the U direction

numvk - number of knots in the V direction

### 8.23.4 BEGIN B-SPLINE SURFACE Command Restrictions

The restrictions of the B-SPLINE SURFACE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The BEGIN B-SPLINE SURFACE command must always be used in conjunction with the END B-SPLINE SURFACE command. The only SIF commands that may appear in a B-spline surface set are the KNOT, POLE, BOUNDARY, and WEIGHT commands. The POLE command must appear within the set.

The SIF B-SPLINE SURFACE command set may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)

### B.23.2 BEGIN B-SPLINE SURFACE Command - Binary Form

The binary form of the BEGIN B-SPLINE SURFACE command is indicated by a command type of 122 in the command header. The subtype field indicates the surface status in the U direction and the values field indicates the surface status in the V direction.

122	7
stype	value
type	
uorder	
vorder	
urule	
vrule	
numuk	
numvk	

stype - flag indicating status in U direction.  
 1 - open  
 2 - closed  
 3 - closed representing a hole

value - flag indicating status in V direction.  
 1 - open  
 2 - closed  
 3 - closed representing a hole

WEI/0.325,0.325,0.65,0.325,0.325,0.325,0.65,0.325,0.325  
 POL/100000,0,0,100000,100000,0,0,100000,0,-100000,100000,0,-100000,0,0,  
 -100000,-100000,0,0,-100000,0,100000,-100000,0,100000,0,0  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/100000,0,0,100000,100000,0,0,100000,0,-100000,100000,0,-100000,0,0,  
 -100000,-100000,0,0,-100000,0,100000,-100000,0,100000,0,0  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/100000,0,-150000,100000,100000,-150000,0,100000,-150000,-100000,  
 100000,-150000,-100000,0,-150000,-100000,-100000,-150000,0,-100000,  
 -150000,100000,-100000,-150000,100000,0,-150000  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/100000,0,-300000,100000,100000,-300000,0,100000,-300000,-100000,  
 100000,-300000,-100000,0,-300000,-100000,-100000,-300000,0,-100000,  
 -300000,100000,-100000,-300000,100000,0,-300000  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/100000,0,-300000,100000,100000,-300000,0,100000,-300000,-100000,  
 100000,-300000,-100000,0,-300000,-100000,-100000,-300000,0,-100000,  
 -300000,100000,-100000,-300000,100000,0,-300000  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/50000,0,-300000,50000,50000,-300000,0,50000,-300000,-50000,50000,  
 -300000,-50000,0,-300000,-50000,-50000,-300000,0,-50000,-300000,  
 50000,-50000,-300000,50000,0,-300000  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 POL/0,0,-300000,0,0,-300000,0,0,-300000,0,0,-300000,0,0,-300000,0,0,  
 -300000,0,0,-300000,0,0,-300000,0,0,-300000  
 WEI/0.5,0.5,1.,0.5,0.5,0.5,1.,0.5,0.5  
 EBS/

- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a BEGIN SYMBOL (BSY/) and an END SYMBOL command (ESY/)
- o immediately before an INCLUDE TEXT command (INC/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o before a CONTINUE command (CON/)
- o between another B-SPLINE SURFACE command set (BBS/ and EBS/)

#### 8.23.5 BEGIN B-SPLINE SURFACE Command Examples

This section presents examples of the SIF B-SPLINE SURFACE command. B-spline surfaces are valid only in 3-D design files. Therefore, examples are given in their 3-D SIF ASCII format along with an IQDS graphic representation of the elements generated.

The following example places a non-uniform, non-rational B-spline surface representing a plane. Figure 8-40 is a graphic illustration of the following SIF ASCII commands:

```

BBS/OU, OV, ET=1, UO=3, VO=3, UR=32, VR=32, UK=6, VK=9
KNO/0. 25, 0. 25, 0. 5, 0. 5, 0. 75, 0. 75, 0. 25, 0. 25, 0. 25, 0. 5, 0. 5, 0. 5, 0. 75, 0. 75,
0. 75
POL/55046, 0, 200000, 55046, 55046, 200000, 0, 55046, 200000, -55046, 55046,
200000, -55046, 0, 200000, -55046, -55046, 200000, 0, -55046, 200000, 55046,
-55046, 200000, 55046, 0, 200000
WEI/0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725
POL/55046, 0, 150000, 55046, 55046, 150000, 0, 55046, 150000, -55046, 55046,
150000, -55046, 0, 150000, -55046, -55046, 150000, 0, -55046, 150000, 55046,
-55046, 150000, 55046, 0, 150000
WEI/0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725
POL/55046, 0, 83486, 55046, 55046, 83486, 0, 55046, 83486, -55046, 55046, 83486,
-55046, 0, 83486, -55046, -55046, 83486, 0, -55046, 83486, 55046, -55046,
83486, 55046, 0, 83486
WEI/0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725
POL/55046, 0, 83486, 55046, 55046, 83486, 0, 55046, 83486, -55046, 55046, 83486,
-55046, 0, 83486, -55046, -55046, 83486, 0, -55046, 83486, 55046, -55046,
83486, 55046, 0, 83486
WEI/0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725, 0. 2725, 0. 545, 0. 2725, 0. 2725
POL/100000, 0, 53846, 100000, 100000, 53846, 0, 100000, 53846, -100000, 100000,
53846, -100000, 0, 53846, -100000, -100000, 53846, 0, -100000, 53846, 100000,
-100000, 53846, 100000, 0, 53846

```

The following example places a B-spline surface as a non-uniform and rational representing a general B-spline surface. Figure B-41 is a graphic representation of the following SIF ASCII commands:

```

BBS/OU, OV, ET=0, UO=3, VO=2, UR=5, VR=5, UK=6, VK=0
KNO/O. 25, 0. 25, 0. 5, 0. 5, 0. 75, 0. 75
BOU/BN=1, 1717986917, 1073741823, 1716689843, 1114602370, 1712803845,
1155298386, 1706344570, 1195666003, 1697338027, 1235542674, 1685820483,
1274767830, 1671838314, 1313183526, 1655447822, 1350635074, 1636715005,
1386971671, 1615715294, 1422047002, 1592533247, 1455719832, 1567262211,
1487854571, 1540003943, 1518321824, 1510868201, 1546998910, 1479972307,
1573770358, 1447440665, 1598528367, 1413404271, 1621173246, 1378000175,
1641613812, 1341370939, 1659767759, 1303664055, 1675561986, 1265031356,
1688932896, 1225628401, 1699826649, 1185613852, 1708199379, 1145148834,
1714017373, 1104396286, 1717257204, 1063520303, 1717905825, 1022685478,
1715960626, 982056239, 1711429439, 941796186, 1704330508, 902067431,
1694692420, 863029948, 1682553984, 824840927, 1667964076, 787654143,
1650981444, 751619332, 1631674473, 716881595, 1610120904, 683580808,
1586407526, 651851061, 1560629825, 621820120, 1532891598, 593608907,
1503304536, 567331020, 1471987777, 543092271, 1439067423, 520990260,
1404676032, 501113984, 1368952085, 483543478, 1332039431, 468349492,
1294086704, 455593207, 1255246725, 445325988, 1215675891, 437589178,
1175533538, 432413930, 1134981306, 429821082, 1094182485, 429821076,
1053301356, 432413911, 1012502533, 437589147, 971950300, 445325945,
931807945, 455593152, 892237107, 468349425, 853397125, 483543400,
815444393, 501113894, 778531734, 520990160, 742807781, 543092161,
708416383, 567330900, 675496021, 593608778, 644179255, 621819981,
614592185, 651850914, 586853948, 683580653, 561076237, 716881433,
537362849, 751619164, 515809270, 787653968, 496502288, 824840748,
479519646, 863029764, 464929726, 902067243, 452791278, 941795995,
443153178, 982056047, 436054235, 1022685284, 431523035, 1063520108,
429577824, 1104396092, 430226433, 1145148641, 433466251, 1185613661,
439284233, 1225628212, 447656952, 1265031170, 458550693, 1303663874,
471921591, 1341370762, 487715806, 1378000004, 505869742, 1413404105,
526310297, 1447440507, 548955166, 1479972156, 573713166, 1510868059,
600484604, 1540003808, 629161682, 1567262086, 659628926, 1592533132,
691763658, 1615715189, 725436480, 1636714910, 760511805, 1655447738,
796848396, 1671838241, 834299940, 1685820422, 872715631, 1697337978,
911940784, 1706344533, 951817452, 1712803821, 992185067, 1716689831,
1032881082, 1717986917, 1073741823
POL/50000, 0, 0, 50000, 50000, 0, 0, 50000, 0, -50000, 50000, 0, -50000, 0, 0, -50000,
-50000, 0, 0, -50000, 0, 50000, -50000, 0, 50000, 0, 0
WEI/O. 5, 0. 5, 1, , 0. 5, 0. 5, 0. 5, 1, , 0. 5, 0. 5
POL/20000, 30000, 100000, 20000, 40000, 100000, 10000, 40000, 100000, 0, 40000,
100000, 0, 30000, 100000, 0, 20000, 100000, 10000, 20000, 100000, 20000, 20000,
100000, 20000, 30000, 100000
WEI/O. 5, 0. 5, 1, , 0. 5, 0. 5, 0. 5, 1, , 0. 5, 0. 5
EBS/

```

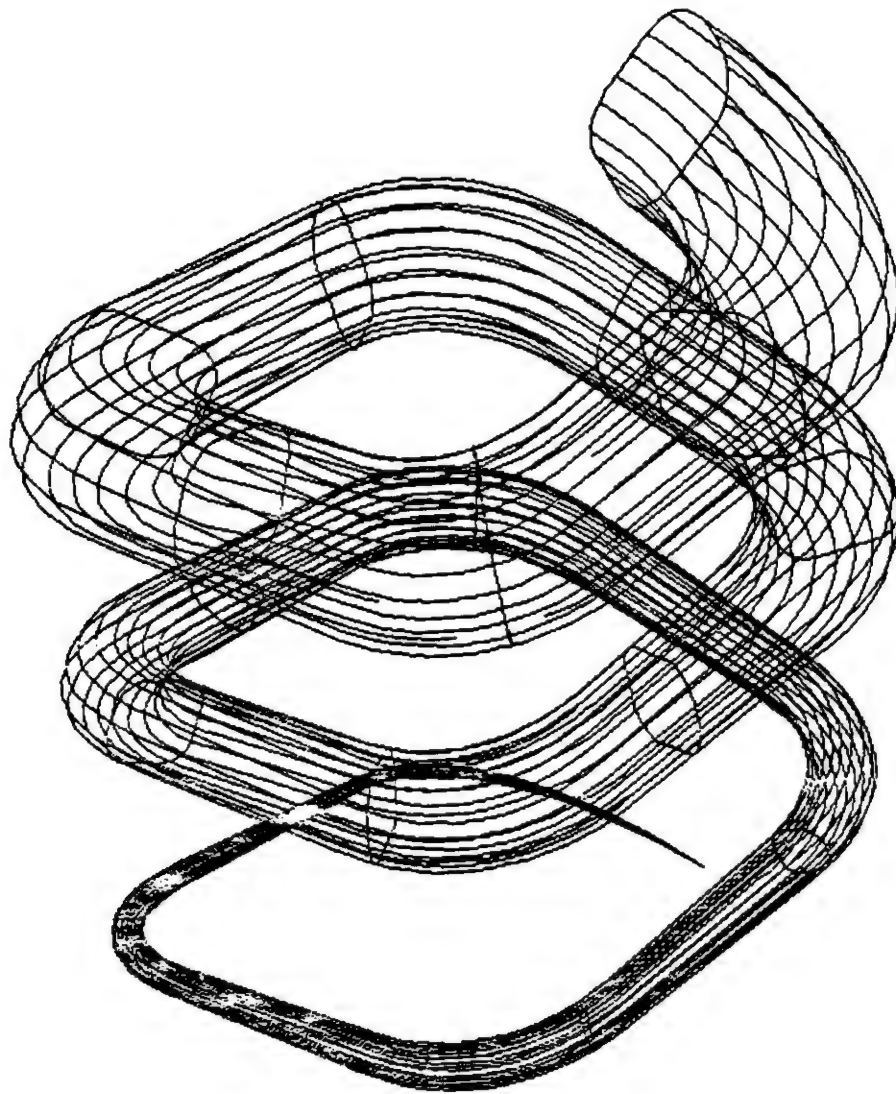


Figure 8-40. Example of B-Spline Surface  
(non-uniform, non-rational)

#### B.24 END B-SPLINE SURFACE Command

The END B-SPLINE SURFACE (EBS) command must be used to indicate the end of the B-spline surface definition. The EBS command may not be used without prior use of a BEGIN B-SPLINE SURFACE (BBS) command.

##### B.24.1 END B-SPLINE SURFACE Command - ASCII Form

The ASCII form of the END B-SPLINE SURFACE command is indicated by a command type of EBS followed by a slash (/):

EBS/

##### B.24.2 END B-SPLINE SURFACE Command - Binary Form

The binary form of the END B-SPLINE SURFACE command is indicated by a command type of 123 in the command header. All other fields are not used.

123	0
0	0

##### B.24.3 END B-SPLINE SURFACE Command - Interface Form

The interface form of the END B-SPLINE SURFACE command allows you to generate a END B-SPLINE SURFACE command by formatting the command and writing it to the SIF output file. The calling sequence is as follows:

CALL CMD123

##### B.24.4 END B-SPLINE SURFACE Command Restrictions

See Section B.23.4.

##### B.24.5 END B-SPLINE SURFACE Command Examples

See Section B.23.5.

#### B.25 B-spline Component Elements

All B-spline curves and surfaces must consist of only four kinds of component elements. These component elements are boundary elements (IGDS element type 25), knot elements (IGDS element type 26), pole elements (IGDS element type 21) and weight elements (IGDS element type 28). The knot, pole, and



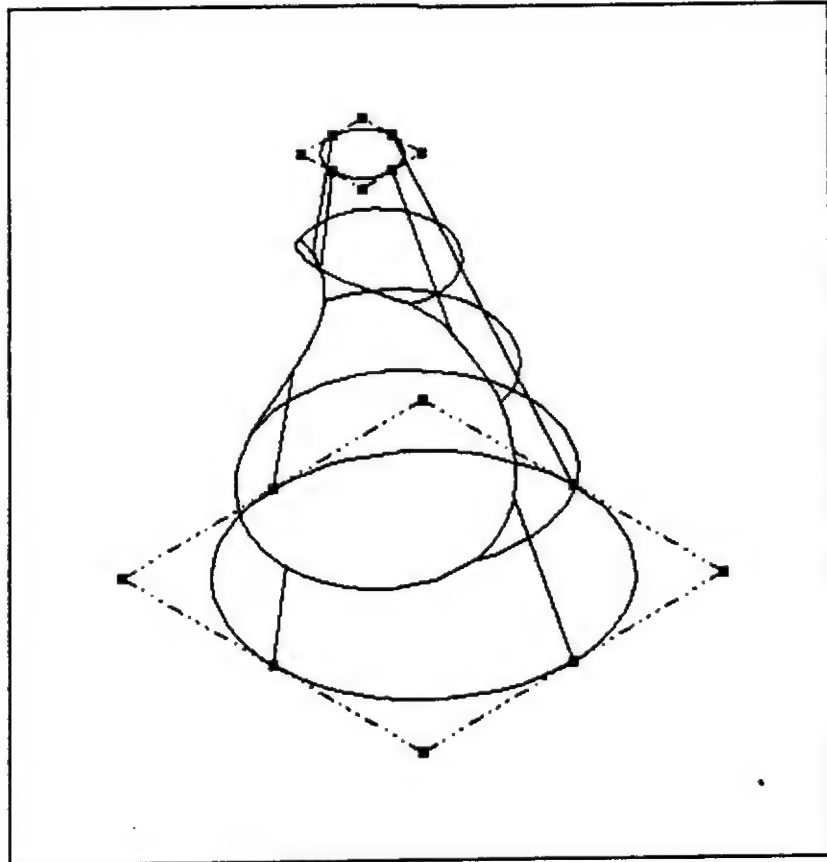


Figure B-41. Example of 3-D SIF B-Spline Surface

### 8.26.3 BOUNDARY Command - Interface Form

The interface form of the BOUNDARY command allows you to generate a BOUNDARY command by specifying the boundary number, the number of vertices in the boundary, and the vertices defining the boundary. The subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

```
CALL CMD125(bnum,nb,bound)
```

bnum - Integer\*4 boundary number.

nb - Integer\*2 number of vertices in boundary.

bound - Integer\*4 vertices defining the boundary.

### 8.27 KNOT Command

The KNOT (KNO) command must be used when defining non-uniform B-spline curves or surfaces. If the KNOT command is used, it must immediately follow the BBC or BBS command. Only one KNO command is allowed in a B-SPLINE CURVE or B-SPLINE SURFACE command set. The format for the KNO command varies depending on whether a B-spline curve or surface is being defined. The B-spline curve has knots specified in the U and V direction. The knot vectors are specified as floating point values between 0 and 1. The number of knots required may be calculated as follows:

```
num_knots = num_poles - order      (open B-spline)
```

```
num_knots = num_poles - 1          (closed B-spline)
```

#### 8.27.1 KNOT Command - ASCII Form

The ASCII form of the KNOT command is indicated by a command type of KNO followed by a slash (/):

```
KNO/k1,k2,...,kn  
KNO/uk1,uk2,...,ukn,vk1,vk2,...,vkn
```

The first form of the command is for B-spline curves. The second form is for B-spline surfaces. The number of knots in the command must equal the number of knots specified in the BBC or BBS command.

weight elements may appear in both 2-D and 3-D SIF ASCII files while the boundary element is found in 3-D B-spline surfaces only. All B-spline curves and surfaces must adhere to very strict component definitions and they must contain the pole element(s). The B-spline component elements must always appear within a SIF B-spline curve set (BBC and EBC) or within a SIF B-spline surface set (BBS and EBS). At no time should B-spline component elements appear standalone within a SIF file.

The following sections describe each of the four B-spline component elements. The SIF ASCII, binary, and interface forms of the component elements are given. For element restrictions and examples, refer to Sections 8.21 and 8.23.

## 8.26 BOUNDARY Command

The BOUNDARY (BOU) command allows you to specify the boundaries of the B-spline surface. The boundary is not specified in three-dimensional space, but as a string of vertices in the UV plane. The BOU command is limited to 151 vertices and must be closed. If more than 151 vertices are required, multiple BOUNDARY commands may be used if you specify the same boundary number for the BOUNDARY commands.

### 8.26.1 BOUNDARY Command - ASCII Form

The ASCII form of the BOUNDARY command is indicated by a command type of BOU followed by a slash (/):

BOU/BN=bnun,u1,v1,u2,v2,...un,vn

The BN keyword defines the boundary number and defaults to 0 if not specified.

### 8.26.2 BOUNDARY Command - Binary Form

The binary form of the BOUNDARY command is indicated by a command type of 125 in the command header. The subtype and value fields are not used.

125	#words
0	0
bnun	
u1	
v1	

8.28 POLE Command

The POLE (POL) command allows you to specify the coordinates of the control polygon of the B-spline curve or surface. The POLE command is required for all B-spline curves and surfaces and must only be used inside the B-SPLINE CURVE and SURFACE command sets. Only one POLE command is allowed in a B-spline curve. Multiple POLE commands are allowed in a B-spline surface, and all POLE commands must have the same number of vertices. The POLE command is limited to 101 vertices with coordinates being specified in UORs.

### 8.28.1 POLE Command - ASCII Form

The ASCII form of the POLE command is indicated by a command type of POL followed by a slash (/):

POL/ $x_1, y_1, x_2, y_2, \dots, x_n, y_n$   
POL/ $x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n$

### 8.28.2 POLE Command - Binary Form

The binary form of the POLE command is indicated by a command type of 126 in the command header. The subtype and value fields are not used.

126	#words
0	0
x1	
y1	

126	#words
0	0
x1	
y1	
z1	

The binary form of the KNOT command is indicated by a command type of 124 in the command header. The subtype and value fields are not used.

### 8.27.3 KNOT Command - Interface Form

The interface form of the KNOT command allows you to generate a KNOT command by specifying the number of knots in the U direction, the array of knots in the U direction, the number of knots in the V direction, and the array of knots in the V direction. The same interface form is used for the B-spline curve except that there are no specified V direction knots. The subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

numuk - Integer\*2 number of knots for the B-spline curve  
or number of knots in the U direction for the B-  
spline surface.

numvk - Integer\*2 number of knots in the V direction defined for the B-spline surface. This argument is ignored for B-spline curves.

B-113

### 8.29.3 WEIGHT Command - Interface Form

The interface form of the WEIGHT command allows you to generate a WEIGHT command by specifying the number and array of weights. The subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

CALL CMD127(nw,weights)

nw - Integer\*2 number of weights. This value must be equal to the number of poles in the previous POLE command.

weights - Real\*4 weight factors.

127	#words
0	0
w1	
*10000.	
w2	
*10000.	
.	.
.	.
.	.
.	.
.	.
.	.

### 8.28.3 POLE Command - Interface Form

The interface form of the POLE command allows you to generate a POLE command by specifying the number of vertices in the pole and the array of vertices defining the pole. The subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

```
CALL CMD126(np,poles)
```

np - Integer\*2 number of poles (101 maximum).

poles - Integer\*4 array of poles.

### 8.29 WEIGHT Command

The WEIGHT (WEI) command allows you to define weight factors for the poles of the control polygon if the B-spline curve or surface is rational. The weights are specified as floating point numbers between 0 and 1. The number of weight factors must be equal to the number of poles in the previous POLE command. The WEIGHT command is recognized only after the POLE command.

#### 8.29.1 WEIGHT Command - ASCII Form

The ASCII form of the WEIGHT command is indicated by a command type of WEI followed by a slash (/):

```
WEI/w1,w2,...wn
```

#### 8.29.2 WEIGHT Command - Binary Form

The binary form of the WEIGHT command is indicated by a command type of 127 in the command header. The subtype and value fields are not used.

### 9.1.1 TEXT LINE Command - ASCII Form

The ASCII form of the TEXT LINE command is indicated by a command type of TXT followed by a slash "/". The slash can then be followed by one of the two ASCII formats for the TEXT LINE command. Examples of the SIF TEXT LINE command are as follows:

```
TXT/MR, TL=tlx, tly, BL=blx, bly, BR=brx, bry,  
MA=t11, t21, t12, t22, text
```

```
TXT/MR, TL=tlx, tly, BL=blx, bly, BR=brx, bry,  
MA=t11, t21, t12, t22, text
```

```
TXT/OR=xo, yo, TH=height, TW=width, AN=angle,  
MA=t11, t21, t12, t22, text
```

The MR keyword indicates that the text string is to be mirrored and is optional. The text string is mirrored about the y-axis relative to the rotation angle. If the MR keyword is not specified, no mirroring occurs.

The TL, BL, and BR keywords indicate the top left, bottom left, and bottom right vertices of the text line in UORs and are required. These keywords can be omitted only if the GENERATE TEXT LINE BLOCK command was used before the TXT/command. The GENERATE TEXT LINE BLOCK command is described in Section 4.1.3.

If using the other ASCII form of the TEXT LINE command, the OR keyword indicates the text line origin in UORs and is required. The TH and TW keywords indicate text height and text width, respectively. These values must be in UORs and are descriptive of a single text character and not the entire text line. The TH and TW keywords are optional and if they are not used, a default value of 100 UORs is given to text height and text width. The AN keyword indicates the angle of rotation of the text line and is optional. The value is in degrees and if the keyword is not specified, a default of 0 degrees is used. This angle is meaningful only if no matrix is present.

Both forms of the ASCII TEXT LINE command may use the optional MA keyword to indicate a transformation matrix. In a 3-D file, this matrix and not the AN keyword should be used to indicate rotation. The matrix should contain four terms for a 2-D file and nine terms for a 3-D file. The range of the terms in the matrix is from -1. to 1.

All characters following the last keyword value are assumed to be the text line characters.



## 9. GRAPHIC TEXT GENERATION COMMANDS

This section describes SIF graphic text generation commands. The description of each command presents the ASCII, binary, and interface forms of the command along with restrictions and examples.

### 9.1 TEXT LINE Command - Generate Text String

The SIF TEXT LINE command is supported in IGDS as a text string. The TEXT LINE command can be used in both 2-D and 3-D SIF files. In 2-D SIF files, the X and Y coordinates of a vertex must be specified while an additional Z-coordinate is necessary in 3-D files. The SIF TEXT LINE command can also have a transformation matrix which defines rotation in the X-Y planes for 2-D text strings and rotation in space for 3-D text strings. The transformation matrix is always row major and must contain four terms for 2-D files and nine terms for 3-D files. It is important to note that the transformation matrix is purely for rotation and not for scaling purposes.

The TEXT LINE command can be used in two forms. One form defines mirroring, the text line range block, and the characters to form the text line. The second form defines the origin, text height and width, the rotation angle, and the text characters. Both of the forms can also have a transformation matrix.

The mirroring field indicates whether the text line is to be mirrored about the Y-axis, relative to the rotation angle, or not mirrored at all. The text is mirrored about the origin of the text line.

The text line range block defines the top left, bottom left, and bottom right vertices of the text line in UORs. The difference between the top left and bottom left vertices defines the text height in UORs. The difference between the bottom left and bottom right vertices and the line length defined in the TEXT LINE CHARACTERISTICS command define the text width in UORs. The angle formed by the X-axis and the line through the bottom left and bottom right vertices defines the rotation angle for the text line.

The second form of the TEXT LINE command requires the origin of the text string in UORs, the text height and text width in UORs, and the rotation angle in degrees measured counterclockwise from the X-axis.

The text field defines the characters to form the text line and is required. The number of characters for a text line may range from 1 to 255.

### 9.1.3 TEXT LINE Command - Interface Form

The interface form allows you to generate a TEXT LINE command by specifying which form of the TEXT command, RANGE or ORIGIN, is used. If the RANGE form is used, the text is generated by specifying the three vertices of a range block to define the height of the text and the length of the entire text string. If the ORIGIN form is used, the interface requires text height, text width, angle of rotation, and text origin. In both forms, the number of characters in the text string, the text string, a mirroring indicator, and a transformation matrix are available. The interface subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

```
CALL CMD60A(topl,botl,botr,nc,text,mflag,matrix,mir,mirpt)
```

```
CALL CMD60B(height,width,rot,origin,nc,text,mflag,matrix,  
mir,mirpt)
```

topl	-	Integer*4 array containing the coordinates of the top left corner of the text line in UORs
botl	-	Integer*4 array containing the coordinates of the bottom left corner of the TEXT LINE command in UORs.
botr	-	Integer*4 array containing the coordinates of the bottom right corner of the text line in UORs
height	-	Integer*4 height of a text character in UORs
width	-	Integer*4 width of a text character in UORs
rot	-	Real*4 rotation angle in degrees (meaningful only if no transformation matrix is given)
origin	-	Integer*4 array containing the coordinate of the TEXT LINE command in UORs
nc	-	Integer*2 number of characters in text line
text	-	byte array containing text line
mflag	-	Integer*2 flag indicating whether a matrix is given
matrix	-	Real*4 array containing the transformation matrix

### 9.1.2 TEXT LINE Command - Binary Form

The binary form of the TEXT LINE command is indicated by a command type of 60 in the command header. The words to follow field must contain the number of Integer\*4 words which are used to complete the definition. The subtype field indicates the RANGE form or the ORIGIN form of the TEXT command and whether a transformation matrix is present. The command value field indicates whether the text is to be mirrored. If the mirroring option is specified, the text is mirrored horizontally about the rotation angle of the text.

60	*WORDS
STYPE	VAL
HEIGHT	
WIDTH	
X0	
Y0	
ROT	*10000.
TEXT	

60	*WORDS
STYPE	VAL
HEIGHT	
WIDTH	
X0	
Y0	
T11	*10000.
T21	*10000.
T12	*10000.
T22	*10000.
TEXT	

60	*WORDS
STYPE	VAL
TLX	
TLY	
BLX	
BLY	
BRX	
BRY	
TEXT	

60	*WORDS
STYPE	VAL
TLX	
TLY	
BLX	
BLY	
BRX	
BRY	
T11	*10000.
T21	*10000.
T12	*10000.
T22	*10000.
TEXT	

stype 0 - RANGE form; no matrix  
           val - 0 - no mirror  
               1 - mirror  
 100 - RANGE form; with matrix   1 - mirror  
       1 - ORIGIN form; no matrix  
 101 - ORIGIN form; with matrix

NOTE: Binary values of the transformation matrix and rotation angle are scaled by a factor of 10000 in order to preserve decimal accuracy. The text field must be blank padded to an even 32-bit word.

The following example places text lines using the range block form of the TEXT LINE command. The text lines are placed at varying justifications with an active angle of zero. Figure 9-2 is a graphic representation of the following SIF ASCII commands.

```
TLC/CO=5, JU=1  
TXT/TL=1000, 1000, BL=-1000, 900, BR=500, 900, JUST1  
TLC/CO=5, JU=2  
TXT/TL=-1000, 50, BL=-1000, -50, BR=-500, -50, JUST2  
TLC/CO=5, JU=3  
TXT/TL=-1000, -900, BL=-1000, -1000, BR=-500, -1000, JUST3
```

mir - Integer\*2 mirror indicator  
 mirpt - reserved for later use

#### 9.1.4 TEXT LINE Command Restrictions

The restrictions of the TEXT LINE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The TEXT LINE command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OFF)
- o before a CONTINUE command (CON/)

#### 9.1.5 TEXT LINE Command Examples

This section presents examples of the SIF TEXT LINE command. Both valid and invalid examples are given and the ASCII form of the command is always used. Valid examples are followed by a figure depicting the IGDS elements which are generated using the ASCII commands.

The following example places text lines using different active angles. The origin form of the TEXT LINE command is used. Figure 9-1 is a graphic representation of the following SIF ASCII commands.

TLC/CD=10, JU=2	
TXT/TH=500, TW=500, AN=0, OR=5000, 4500,	TEXT
TXT/TH=500, TW=500, AN=45, OR=5000, 4500,	MAY
TXT/TH=500, TW=500, AN=90, OR=5000, 4500,	BE
TXT/TH=500, TW=500, AN=135, OR=5000, 4500,	PLACED
TXT/TH=500, TW=500, AN=180, OR=5000, 4500,	AT
TXT/TH=500, TW=500, AN=225, OR=5000, 4500,	ANY
TXT/TH=500, TW=500, AN=270, OR=5000, 4500,	ANGLE
TXT/TH=500, TW=500, AN=315, OR=5000, 4500,	DESIRED

The following is an example of text placing into a 3-D file. Figure 9-3 is a graphic representation of the following SIF ASCII commands.

```
TLC/CO=26, JU=3
TXT/TL=46059, 537962, 0, BL=46059, 533403, 0, BR=578993, 533403,
    0, ABCDEFGHIJKLMNOPQRSTUVWXYZ
TLC/CO=10, JU=3
TXT/TL=46059, 528844, 0, BL=460459, 524285, 0, BR=506049,
    524285, 0, 1234567890
TLC/CO=18, JU=3
TXT/TL=460459, 519726, 0, BL=460459, 515167, 0, BR=542521,
    515167, 0, !@#$%^&*( )*:, +, . ? /
TXT/TL=508753, 463682, 0, BL=511976, 460459, 0, BR=563543,
    512027, 0, TEXT AT AN ANGLE
```

The following is an example of an illegal usage of the TEXT RANGE command since the top left vertex and bottom left vertex defines the text as having no height.

```
TLC/CO , JU=3
TXT/TL=1000, 500, BL=1000, 500, BR=2000, 500, TEXT WITH NO
    HEIGHT
```

The following is an example of using different text fonts when placing text strings. Figure 9-4 is a graphic representation of the following SIF ASCII commands.

```
TLC/CO=9, JU=4
FNT/7
TXT/TH=1500, TW=1500, OR=5000, 7000, Different
TLC/CO=10, JU=4
FNT/23
TXT/TH=1000, TW=1000, OR=5000, 5000, text fonts
TLC/CO=11, JU=4
FNT/26
TXT/TH=1000, TW=1000, OR=5000, 3500, may be used
```

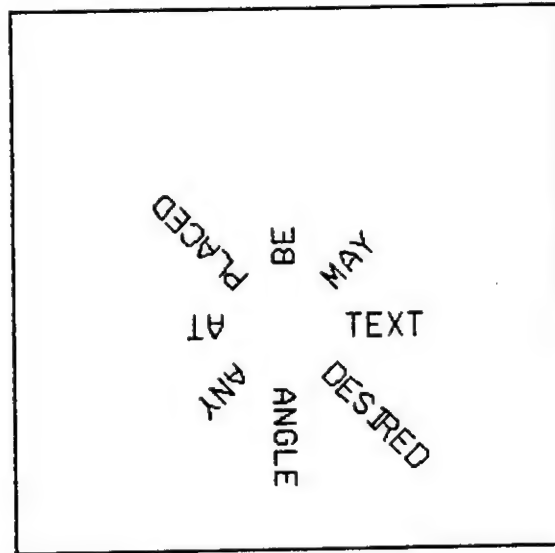


Figure 9-1. Example of SIF Text Lines at Varying Angles

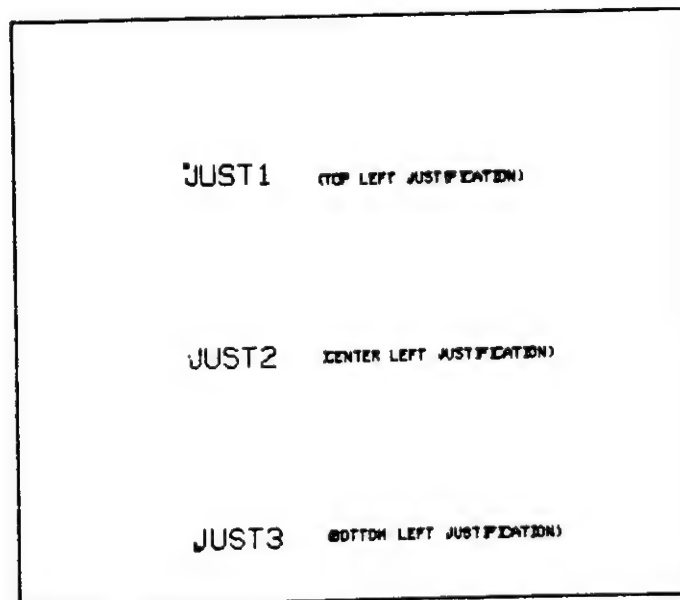


Figure 9-2. Example of SIF Text Lines at Varying Justifications

## 9.2 GENERATE PARAGRAPH Command - Generate Text Node

The SIF GENERATE PARAGRAPH command set is supported in IGDS as a text node. The GENERATE PARAGRAPH command can be used in both 2-D and 3-D SIF files. In 2-D SIF files, the x and y coordinates of a vertex must be specified while an additional z coordinate is necessary in 3-D files. The SIF GENERATE PARAGRAPH command can also have a transformation matrix which defines rotation in the x-y plane for 2-D text nodes and rotation in space for 3-D text nodes. The transformation matrix is always row major and must contain four terms for 2-D files and nine terms for 3-D files.

NOTE: The transformation matrix is purely for rotation and not for scaling purposes.

The row major matrix should always be input as follows:

$$\begin{bmatrix} t11 & t12 & t13 \\ t21 & t22 & t23 \\ t31 & t32 & t33 \end{bmatrix}$$

The GENERATE PARAGRAPH command can be used in two forms. One form defines mirroring, the text node range block, the displayable attribute number, and the text node number. The second form defines the displayable attribute number, text node number, origin, text height, text width and rotation angle. Both of the forms can also have a transformation matrix.

If the GENERATE PARAGRAPH command is used, it must be followed by GENERATE PARAGRAPH LINE command(s) and a CLOSE PARAGRAPH command. An empty text node may be created by omitting the GENERATE PARAGRAPH LINE command(s).

The mirroring field indicates whether the text node is to be mirrored. If it is mirrored, it is mirrored about the Y-axis relative to the rotation angle of the text node. This is an optional keyword.

The displayable attribute number indicates whether the text node is to be a displayable attribute and is optional. The displayable attribute number has a range of 0 to 255 and defaults to 0 if not specified. A value of 0 indicates that the text node is not a displayable attribute. If a displayable attribute number is specified, you must have previously defined an entity to which the text node can be linked.

The text node number indicates a unique number assigned to each text node and is optional. The text node number has a range of -1 to 65535 and defaults to -1 if not specified. A value of -1 indicates that the next available text node number is to be used and is recommended for the creation of the graphic file.



ABCDEFGHIJKLMNOPQRSTUVWXYZ  
1234567890  
!@\*%&'&=(>||+,.?/

Figure 9-3. Example of SIF 3-D Text Lines

**Different**  
*text fonts*  
may be used

Figure 9-4. Example of SIF Text Lines with Varying Fonts

The TL, BL, and BR keywords indicate the top left, bottom left, and bottom right vertices of the text node in UORs and is required.

The TH and TW keywords indicate text height and text width, respectively. These values must be in UORs and are descriptive of a single text character and not the entire text node. The TH and TW keywords are optional and if they are not used, a default of 100 UORs is assigned to text height and text width.

The AN keyword indicates the angle of rotation of the text node and is optional. The angle can range from 0 to 360 degrees with five-digit decimal accuracy available and is measured counterclockwise from the X-axis. If not specified, a default of 0 degrees is used. The angle is meaningful only if the transformation matrix is present.

Both forms of the GENERATE PARAGRAPH command can use the optional MA keyword to indicate a transformation matrix. In a 3-D file, this matrix and not the AN keyword should be used to indicate rotation. The matrix is row major and should contain four terms for a 2-D file and nine terms for a 3-D file. The terms may have up to five-digit decimal accuracy. If the MA keyword is not specified, an identity matrix is assumed. The range of the terms of the matrix is from -1. to 1.

The text node range block defines the top left, bottom left, and bottom right vertices of the text node in UORs. The difference between the top left and bottom left vertices and the number of lines and line spacing defined in the PARAGRAPH CHARACTERISTICS command defines the text height in UORs. The difference between the top left and bottom right vertices and the line length defined in the PARAGRAPH CHARACTERISTICS command defines the text width in UORs. The angle formed by the X-axis and the line through the bottom left and bottom right vertices define the rotation angle for the text node.

The second form, the ORIGIN form, of the GENERATE PARAGRAPH command requires the origin of the text node in UORs and is required. The text height and text width keywords indicate height and width of individual characters within the text node and not the height and width of the entire text node. If not specified, text height and text width default to 100 UORs. The paragraph angle indicates the rotation of the text node measured counterclockwise from the X-axis. If not specified, this angle is 0 degrees. The angle has a range of 0 to 360 degrees and can be entered with up to five-digit decimal accuracy.

#### 9.2.1 GENERATE PARAGRAPH Command - ASCII Form

The ASCII form of the GENERATE PARAGRAPH command is indicated by a command type of PAR followed by a slash "/". The slash can then be followed by one of the two ASCII formats for the GENERATE PARAGRAPH command. Examples of the GENERATE PARAGRAPH command are as follows:

```
PAR/MR, FO=dap, ID=id, TL=tlx, tly, BL=blx, bly, BR=brx, bry,  
MA=t11, t21, t12, t22
```

```
PAR/MR, FO=dap, ID=id, OR=xo, yo, TH=height, TW=width, AN=angle,  
MA=t11, t21, t12, t22
```

The MR keyword indicates that the text node is to be mirrored and is optional. The text node is mirrored about the Y-axis relative to the rotation angle. If the MR keyword is not present, no mirroring occurs.

The FO keyword indicates the text node displayable attribute number and is optional. If not specified, "dap" is assumed to be 0 which indicates that the text node is not a displayable attribute.

The ID keyword indicates the text node number and is optional. If not specified, "id" is assumed to be -1 which indicates that the next available text node number is to be used.

### 9.2.3 GENERATE PARAGRAPH Command - Interface Form

The interface form allows you to generate a GENERATE PARAGRAPH command by specifying which form of the GENERATE PARAGRAPH command, text or range, is used. If the range form is used, the text node is generated by specifying the three vertices of a range block to define the height of the text node and the length of the longest text string. If the origin form is used, the interface requires text height and width of a single text character, angle of rotation, and text node origin. In both forms of the GENERATE PARAGRAPH command, a mirroring indicator, matrix indicator, and transformation matrix are available. The interface subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:

```
CALL CMD61A(dap,id,topl,botl,botr,mflag,matrix,mir,mirpt)
```

```
CALL CMD61B(dap,id,height,width,rot,origin,mflag,matrix,  
            mir, mirpt)
```

dap - Integer\*2 displayable attribute number

id - Integer\*4 text node number

topl - Integer\*4 array containing the coordinates of the top left corner of the text node in UORs

botl - Integer\*4 array containing the coordinates of the bottom left corner of the text node in UORs.

botr - Integer\*4 array containing the coordinates of the bottom right corner of the text node in UORs

mflag - Integer\*2 flag indicating whether a matrix is given (0-no matrix, 1-matrix)

matrix- Real\*4 array containing the transformation matrix

mir - Integer\*2 mirror indicator  
0 - no mirroring  
1 - mirroring vertically relative to rotation angle about origin

mirpt - Reserved for later use

height- Integer\*4 height of text character in UORs

width - Integer\*4 width of text character in UORs

### 9.2.2 GENERATE PARAGRAPH Command - Binary Form

The binary form of the GENERATE PARAGRAPH command is indicated by a command type of 61 in the command header. The words to follow field must contain the number of Integer\*4 words which are used to complete the definition. The subtype field indicates the range form or the origin form of the GENERATE PARAGRAPH command and whether a transformation matrix is present. The command value field indicates whether the text node is to be mirrored. If the mirroring option is specified, the text node is mirrored vertically about the rotation angle of the text node.

61	*WORDS
STYPE	VAL
DAP	-
ID	-
HEIGHT	-
WIDTH	-
X0	-
Y0	-
ROT	-
*10000.	-

61	*WORDS
STYPE	VAL
DAP	-
D	-
HEIGHT	-
WIDTH	-
X0	-
Y0	-
T11	-
*10000.	-
T21	-
*10000.	-
T12	-
*10000.	-
T22	-
*10000.	-

61	*WORDS
STYPE	VAL
DAP	-
D	-
TLX	-
TLY	-
BLX	-
BLY	-
BRX	-
BRY	-

61	*WORDS
STYPE	VAL
DAP	-
D	-
TLX	-
TLY	-
BLX	-
BLY	-
BRX	-
BRY	-
T11	-
*10000.	-
T21	-
*10000.	-
T12	-
*10000.	-
T22	-
*10000.	-

stype      0 - range form; no matrix    val  
             0 - no mirror  
             1 - mirror  
          100 - range form; with matrix  
             1 - origin form; no matrix  
          101 - origin form; with matrix

NOTE:      Binary values of the transformation matrix are scaled by a factor of 10000 in order to preserve decimal accuracy.

The following example places a text node with one line at an angle of 90 degrees. The range form is used with a matrix. The number of text lines and the number of characters in the text line dictate that the text node is placed with a text height of 200 UORs and a text width of 100 UORs. This is text node number 10 and the displayable attribute number is 15. Figure 9-6 is a graphic representation of the following SIF ASCII commands.

```
PAR/FO=15, ID=10, TL=200, 200, BL=0, 200, BR=0, 1300, MA=1., 0.,  
0., 1  
PLN/SAVE OUR SHIP  
CLP/
```

### 9.3 GENERATE PARAGRAPH LINE Command - Text Node Line

The SIF GENERATE PARAGRAPH LINE command is supported in IGDS as a text node line. If GENERATE PARAGRAPH LINE command(s) are used, they must be immediately preceded by a GENERATE PARAGRAPH command and immediately followed by a CLOSE PARAGRAPH command. The PARAGRAPH LINE command defines the nonedit delimiter, the edit delimiter, and characters for the text node line.

The nonedit delimiter indicates the ASCII character to enclose nonedit fields and is optional. Any ASCII character which does not appear on the text node line is valid. Nonedit fields are those which are not included as enter data fields. If the nonedit delimiter and the edit delimiter are not specified, the entire text field is assumed to be noneditable text.

The edit delimiter indicates the ASCII character to enclose edit fields and is optional. Any ASCII character which does not appear on the text node line is valid. Edit fields are those which are included as enter data fields. If a displayable attribute number was specified in the GENERATE PARAGRAPH command, enter data field(s) should appear in some GENERATE PARAGRAPH LINE commands.

rot - Real\*4 rotation angle in degrees (meaningful only if no transformation matrix is given)

origin- Integer\*4 array containing the coordinates of the text node origin in UORs

#### 9.2.4 GENERATE PARAGRAPH Command Restrictions

The restrictions of this command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The GENERATE PARAGRAPH command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES COMMAND (ASV/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OFF)
- o between a TEXT STRING command (TXT/) and its corresponding INCLUDE command (INC/)
- o between a SYMBOL (SYM/) or END SYMBOL command (ESY/) and its corresponding INCLUDE command (INC/)
- o before a CONTINUE command (CON/)

The GENERATE PARAGRAPH command must always have a corresponding CLOSE PARAGRAPH command (CLP/)

#### 9.2.5 PARAGRAPH Command Examples

This section presents examples of the SIF GENERATE PARAGRAPH command. Both valid and invalid forms are given and the ASCII form of the command is always used.

The following example places a text node at 45 degrees with three lines using the origin form. Text height is 175 UORs while text width defaults to 100 UORs. Figure 9-5 is a graphic representation of the following SIF ASCII commands.

```
PAR/OR=100,100,AN=45,TH=175
PLN/THIS IS A SIMPLE TEXT
PLN/NODE PLACED AT AN ANGLE
PLN/OFF 45 degrees
CLP/
```

The text field indicates the characters to be placed on the text node line and must include any required delimiters. A text node line can contain a maximum of 255 characters excluding delimiters. A text node can contain a maximum of 454 characters in a 2-D file and 436 characters in a 3-D file.

### 9.3.1 GENERATE PARAGRAPH LINE Command - ASCII Form

PLN/NE=noned,ED=edit,text

The NE keyword indicates the nonedit delimiter in the text field and is optional. If specified, all characters enclosed by two nonedit delimiters are not included in the text node line as an enter data field. If not specified and the ED keyword is also not specified, the entire text field is assumed to be noneditable text.

The ED keyword indicates the edit delimiter in the text field and is optional. If specified, all characters enclosed by two edit delimiters are included in the text node line as enter data fields.

The text field defines the text to be placed on the text node line and includes any required delimiters.

### 9.3.2 GENERATE PARAGRAPH LINE Command - Binary Form

The binary form of the GENERATE PARAGRAPH LINE command is indicated by a command type of 62 in the command header. The words to follow field must contain the number of Integer\*4 words needed to define the paragraph line. The subtype field indicates the nonedit delimiter and the command value field indicates the edit delimiter. The paragraph line itself constitutes the remaining Integer\*4 words of the paragraph line definition.

If "noned" and "edit" are 0 or blank, the entire text field is assumed to be noneditable text. The text field must be blank padded to an even 32-bit word.

62	*WORDS
NONE0	EDIT

TEXT

### 9.3.3 GENERATE PARAGRAPH LINE Command - Interface Form

The interface form allows you to generate a GENERATE PARAGRAPH LINE command by specifying edit and nonedit delimiters, the number of characters in the paragraph line, and the paragraph line itself. The interface subroutine formats the command and writes it to a SIF output file. The calling sequence is as follows:





Figure 9-5. Example of Text Node at 45 Degrees

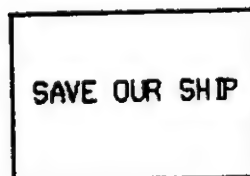


Figure 9-6. Example of SIF Text Node at 90 Degrees

THIS LINE HAS 1 ENTER DATA FIELDS)  
BUT THIS LINE HAS NO ENTER DATA FIELDS

Figure 9-7. Example of SIF Enter Data Field Within Text Node

The following paragraph set is an example of an illegal use of the paragraph line and text string:

PAR/ID=6,FO=3,OR=0,0,0,MA=1.0.,0.,0.,1.,0.,0.,0.,1.  
PLN/3-D TEXT NODES ARE AVAILABLE  
TXT/THIS LINE IS ILLEGAL WITHIN A PARAGRAPH command SET  
CLP/

The following example shows how to place an empty text node. For all values not specified, defaults are assumed.

PAR/OR=100,40  
CLP/

The following example is an illegal usage of the CLOSE PARAGRAPH command:

TXT/OR=-10,80,TH=50,TW=25,FOLLOW ME  
CLP/  
CIR/RA=50,OR=1000,0  
PAR/OR=70,70  
PLN/TEST CASE FOR TEXT NODE  
LST/OP,10,10,50,40,90,70

CALL CMD62(noned, edit, nc, text)

noned - byte character denoting nonedit delimiter  
edit - byte character denoting edit delimiter  
nc - Integer\*2 number of characters in "text" array  
text - byte array containing the characters to be placed  
on the text node line and any required delimiters

#### 9.3.4 GENERATE PARAGRAPH LINE Command Restrictions

The GENERATE PARAGRAPH LINE command can only be placed between a GENERATE PARAGRAPH command (PAR/) and a CLOSE PARAGRAPH command (CLP/). Restrictions on placement of the GENERATE PARAGRAPH command are found in Section 9.2.5.

#### 9.3.5 GENERATE PARAGRAPH LINE Command Examples

This section presents examples of the GENERATE PARAGRAPH LINE command using the ASCII format. Both valid and invalid examples are given. Figure 9-7 is a graphic representation of the following SIF ASCII commands.

```
PAR/TL=100,225,BL=100,0,BR=3500,0
PLN/NE=!,ED=?,!THIS LINE HAS!?1?!ENTER DATA FIELD!
PLN/BUT THIS LINE HAS NO ENTER DATA FIELDS
CLP/
```

TPC/NU=4, SP=25, CO=35, JU=5  
PA3/OR=100, 100, AN=135. , TH=100  
PAR/  
PLN/NOTICE THAT INFORMATION DOES NOT  
PLN/HAVE TO APPEAR WITH THE PARAGRAPH  
PLN/COMMAND LINE BUT IT MUST STILL  
PLN/BE PRESENT  
CLP/

#### 9.4 CLOSE PARAGRAPH - End Text Node

The SIF CLOSE PARAGRAPH command must be used to indicate the end of a text node definition. The CLOSE PARAGRAPH command may not be used without prior use of a PARAGRAPH command. (See Section 9.2.)

##### 9.4.1 CLOSE PARAGRAPH - ASCII Form

The ASCII form of the CLOSE PARAGRAPH command is indicated by a command type of CLP followed by a slash "/".

##### 9.4.2 CLOSE PARAGRAPH - Binary Form

The binary form of the CLOSE PARAGRAPH command is indicated by a command type of 63 in the command header. All other fields are zero.

63	0
0	0

##### 9.4.3 CLOSE PARAGRAPH - Interface Form

The interface form allows you to generate a CLOSE PARAGRAPH command which is formatted and written to a SIF output file. The calling sequence is as follows:

CALL CMD63

##### 9.4.4 CLOSE PARAGRAPH Command Restrictions

The CLOSE PARAGRAPH command may appear only in conjunction with a GENERATE PARAGRAPH command (see Section 9.2), therefore, all restrictions governing the GENERATE PARAGRAPH command apply to the CLOSE PARAGRAPH command.

##### 9.4.5 CLOSE PARAGRAPH Command Examples

This section provides examples of the CLOSE PARAGRAPH command. Both valid and invalid forms are given and the ASCII form is always used.

The following example shows that to define a text node, a PAR/ and CLP/ command set must be specified. Figure 9-8 is a graphic representation of the following SIF ASCII commands.

## NOTES

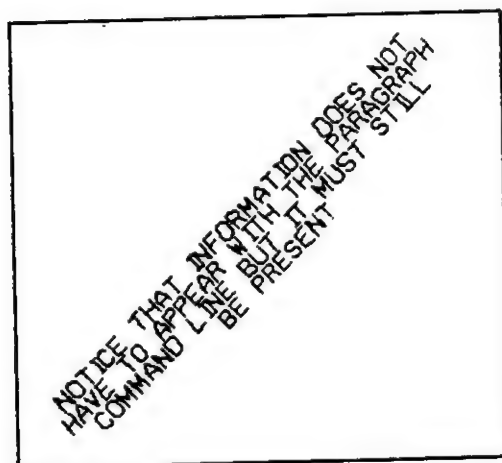


Figure 9-8. Example of SIF Text Node Using PA3/ Command

#### 10.1.4 PAD Command Restrictions

The restrictions of the PAD command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The PAD command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OFF).
- o before a CONTINUE command (CON/)

#### 10.1.5 PAD Command Examples

An example of the SIF ASCII PAD command is as follows:

PAD/12

#### 10.2 GRAPHIC ASSOCIATION DESCRIPTOR Command - Group Data

The SIF GRAPHIC ASSOCIATION DESCRIPTOR command is supported in IGDS as a group data element. The graphic association descriptor defines the graphic group number and a description for the graphically grouped elements.

The graphic group number has a range -1 to 65535 and is required. Graphic group numbers greater than 65535 are changed to that number module 65535. The initial value for the graphic group number is 0. If a value of -1 is used, the current graphic group number is used. This mode is recommended and requires that the ASSOCIATION command (see Section 7.4) be used before the GRAPHIC ASSOCIATION DESCRIPTOR command.

The description is a free field ASCII description of the graphically grouped elements and is required. The description has a range of 1 to 1,000 characters.

In the following subsections, ggnum represents the graphic group number, and "desc" represents the description.



## 10. MISCELLANEOUS COMMANDS

This section describes SIF miscellaneous commands. Miscellaneous commands are those which do not fit neatly into the other sections. The description of each command presents the ASCII, binary, and interface forms of the command.

### 10.1 PAD - No Operation

The SIF PAD command is included as a no operation command. The PAD command is ignored by the SIF translators. The PAD command can be used if you do not wish commands to cross record boundaries in a SIF binary file. In the following subsections, "nwords" are used to indicate the number of 32 bit words to be padded with binary zeroes.

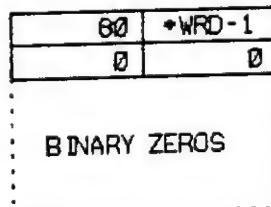
#### 10.1.1 PAD Command - ASCII Form

The ASCII form of the PAD command is indicated by a command type of PAD followed by a slash "/". The slash is then followed by the number of 32-bit words to be padded. An example of the SIF PAD command is as follows:

PAD/32

#### 10.1.2 PAD Command - Binary Form

The binary form of the PAD command is indicated by a command type of 80 in the command header. The number of words to follow field contains the number of Integer\*4 words to be padded with binary zeroes.



#### 10.1.3 PAD Command - Interface Form

The interface form allows you to generate a PAD command by specifying the number of words to be padded. The interface subroutine formats the command and writes it to a SIF output file. The following is an example of the calling sequence used when generating a SIF PAD command using the interface form:

CALL CMD80(nwords)

nwords - Integer\*2 number of 32-bit words to be padded

#### 10.2.4 GRAPHIC ASSOCIATION DESCRIPTOR Command Restrictions

The restrictions of the GRAPHICS ASSOCIATION DESCRIPTOR command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The graphic association descriptor may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a PATTERNING ON and a PATTERNING OFF command (PTN/OF)
- o before a CONTINUE command (CON/)

#### 10.2.5 GRAPHIC ASSOCIATION DESCRIPTOR Command Examples

This section presents examples of the SIF GRAPHIC ASSOCIATION DESCRIPTOR command. Both valid and invalid examples are given and the ASCII form of the command is always used.

The following is an example of the GRAPHIC ASSOCIATION DESCRIPTOR command using graphic group number 500 and an ASCII description.

GAD/SN=500,-10.3,-10.4,-10.6.-scale factors

The following is an illegal use of the GRAPHIC ASSOCIATION DESCRIPTOR because a graphic group number is not specified:

GAD/DON'T TRY THIS METHOD

#### 10.3 CONTINUE Command

The SIF CONTINUE command can continue the SIF LINE STRING, CURVE, CONIC, or ASSOCIATIVE VALUES command. You can specify any number of CONTINUE commands. You are allowed to define coordinate data or attribute data depending on what type of command is being continued.

### 10.2.1 GRAPHIC ASSOCIATION DESCRIPTOR Command - ASCII Form

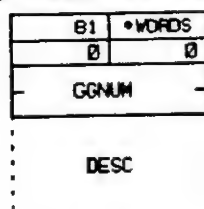
The ASCII form of the GRAPHIC ASSOCIATION DESCRIPTOR command is indicated by a command type of GAD followed by a slash "/". The slash is then followed by the graphic group number and the description. An example of the SIF graphic association descriptor is as follows:

GAD/SN=ggnum, desc

The SN keyword indicates the graphic group number and is required. The description, in ASCII, is also required and must follow the graphic group number specification. The graphic group number must have a range of -1 to 65535 and the description must be between 1 and 1000 characters.

### 10.2.2 GRAPHIC ASSOCIATION DESCRIPTOR Command - Binary Form

The binary form of the GRAPHIC ASSOCIATION DESCRIPTOR command is indicated by a command type of 81 in the command header. The words to follow field must contain the number of Integer\*4 words which are used to complete the definition.



ggnum - graphic group number

desc - the description field must be padded to an even 32-bit word.

### 10.2.3 GRAPHIC ASSOCIATION DESCRIPTOR Command - Interface Form

The interface form allows you to generate a GRAPHIC ASSOCIATION DESCRIPTOR command by specifying a graphic group number, number of characters in description, and the description. The interface subroutine formats the command and writes it to a SIF output file. The following example describes the calling sequence that is required when generating a SIF GRAPHIC ASSOCIATION DESCRIPTOR command using the interface form:

CALL CMD81(ggnum, nc, desc)

ggnum - Integer\*4 graphic group number

nc - Integer\*2 number of characters in the graphic group description

desc - byte array containing the graphic group description

CALL CMD82(nval, flag, array)

nval - Integer\*2 number of values in array. nval is the number of bytes if continuing an ASSOCIATIVE VALUES command and the number of 32-bit coordinates if continuing a line string or curve.

flag - Integer\*2 flag denoting continue type

1 - continuing byte data

4 - continuing coordinate data

array - byte array containing continued associative values if flag is 1 and an Integer\*4 array containing continued coordinates if flag is 4.

If the interface library is being used, SIF CONTINUE commands are automatically generated where necessary and CMD82 need not be called.

#### 10.3.4 CONTINUE Command Restrictions

The restrictions of the CONTINUE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The CONTINUE command may appear in the SIF file in the following conditions:

- o immediately after a LINE STRING (LST/), a CURVE (CUR/) or a CONIC command (CNC/)
- o immediately after an ASSOCIATIVE VALUES command (ASV/)
- o immediately after another CONTINUE command (CON/)

#### 10.3.5 CONTINUE Command Examples

This section presents examples of the SIF CONTINUE command. Both valid and invalid examples are given and the ASCII form of the command is always used.

The following is an example of the CONTINUE command used in conjunction with a 3-D CONIC command.

```
CNC/0, 100, 100, 0, 125, 521, 0, 175, 571, 0, 200, 984,  
    0, 110, 110, 0, 135, 531, 0, 185, 581, 0, 210, 994  
CON/0, 120, 120, 0, 145, 631
```

If a line string, curve or conic is being continued, the CONTINUE command must contain coordinate data. The CONTINUE command must begin with an x-coordinate. In a 3-D SIF file, a z-coordinate must be added to the x-y coordinate pairs.

If an ASSOCIATIVE VALUES command is being continued, the CONTINUE command can begin with any character in the attribute values field.

In the following subsections, "vert" represents continued vertices, and "values" represents continued attribute data.

### 10.3.1 CONTINUE Command - ASCII Form

The ASCII form of the CONTINUE command is indicated by a command type of CON followed by a slash "/". The slash is then followed by either vertices or attribute data. Examples of the SIF CONTINUE command are as follows:

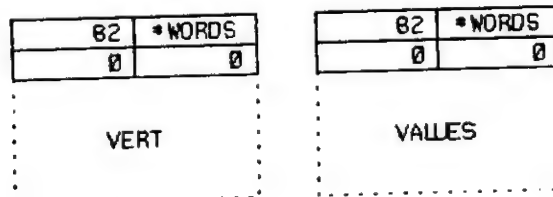
CON/VERT

CON/VALUES

The SIF ASCII CONTINUE command is not required if the SIF ASCII continuation line is used. An ASCII continuation line is indicated by blanks in the first four columns of the text record. Any number of continuation lines can be used.

### 10.3.2 CONTINUE Command - Binary Form

The binary form of the CONTINUE command is indicated by a command type of 82 in the command header. The number of words to follow field contains the number of Integer\*4 words needed to complete the command definition.



### 10.3.3 CONTINUE Command - Interface Form

The interface form allows you to generate a CONTINUE command by specifying the number of values being passed, the value type which is either coordinate data or attribute values, and the array containing the values. The interface subroutine formats the command and writes it to a SIF output file. The following is an example of the calling sequence used when generating a SIF CONTINUE command using the interface form:

The attribute data field indicates attribute names, numbers, and values to be inserted and is optional. If omitted, an entity with all default values is inserted. If specified, the field must begin and end with the specified delimiter character. The data after the odd delimiters contains the attribute name or number. The attribute name or number can consist of the characters A-Z and 0-9. If the field contains all numeric characters, it is assumed to be an attribute number. If the field contains one or more alpha characters, it is assumed to be an attribute name. All blank characters in the name or number field are ignored. The data after the even delimiters contains the attribute values. The textual form is supported for all attribute types. For coded attributes, the code number can be used by enclosing the code number with the pound sign "#" (#1#).

In the following subsections, "delim" represents the delimiter character, "pent" represents the parent entity number, "pocc" represents the parent occurrence number, and "values" represents the attribute data.

#### 10.4.1 ASSOCIATIVE VALUES Command - ASCII Form

The ASCII form of the ASSOCIATIVE VALUES command is indicated by a command type of ASV followed by a slash "/". The slash is then followed by the necessary keywords and attribute data. An example of the SIF ASSOCIATIVE VALUES command is as follows:

```
ASV/DE=delim,ID=pent,KE=pocc,values
```

The DE keyword indicates the delimiter character to separate attribute data in the "values" field and is required if attribute data is specified.

The ID keyword indicates the parent entity number and is optional. If omitted, the entity is inserted without parent association.

The KE keyword indicates the parent occurrence number and is optional. This keyword should not be used since it requires extensive knowledge of the database.

The values field indicates attribute data and is optional. If omitted, an entity with all default values is inserted.

The following is an example of a CONTINUE command used with a 2-D line string.

```
LST/OP, 100, 100, 200, 200,  
      300, 300, 400, 400  
CON/500, 500, 600, 600, 700, 700  
CON/800, 800
```

The following is an example of a CONTINUE command being used illegally:

```
TXT/OR=100, 0, TH=715, TW=500, TEST CASE  
CON/100, 0, 10, -100, 0, -200
```

The following is an example of a CONTINUE command to be used in conjunction with an ASSOCIATIVE VALUES command:

```
ASV/DE=?ID=2, ?! ?HOUSE BOAT?2?REFRIGERATOR?3?  
CON/NORTH-BY-NORTHWEST?4?ANDERSON ST?5?35805?  
      6?PUBLIC SCHOOL??S?8?157 S CONGER BLVD?  
CON/?REC CENTER?10?POST OFFICE?
```

#### 10.4 ASSOCIATIVE VALUES Command - Define Attribute Data

The SIF ASSOCIATIVE VALUES command can define attribute data for the new entity to be inserted into the database. The ASSOCIATIVE VALUES command must be immediately preceded by a SIF IDENTIFIER command to define the new entity. (See Section 7.8.) You are allowed to define the delimiter for attribute data, the parent entity number, the parent occurrence number, and the attribute data.

The delimiter character separates attribute names or numbers and attribute values and is required if attribute data is specified. Any ASCII character which is not used in the attribute names, numbers, or values is valid.

The parent entity number indicates the entity number in the database which is designated to be the parent entity of the entity currently being defined and is optional. If omitted, the entity is inserted without parent association.

The parent occurrence number indicates the relative position of the parent entity in the database and is optional. If omitted and the parent entity number is specified, the occurrence number of the latest entity inserted with an entity number equal to the parent entity specified will be used. Occurrence numbers have a range of 0 to 1,048,575. A value of 0 is recommended since you may not be familiar with the contents of an existing database.

nc - Integer\*2 number of characters in the values array. A value of 0 indicates that all default attribute values are desired. A value of -1 indicates that the utility subroutine SIFATT will be used.

values - Byte array containing attribute data separated by delimiters.

The interface utility subroutine SIFATT (see Section 6.5.1) is provided to allow you to specify data for each attribute with separate calls. By using SIFATT, you do not have to build the values array with delimiters. If SIFATT is used, CMD83 must be initially called as follows:

```
CALL CMD83 (delim, pent, pocc, -1, 0)
```

#### 10.4.4 ASSOCIATIVE VALUES Command Restrictions

This section explains restrictions on the use of the SIF ASSOCIATIVE VALUES command. For a complete description of conflicting commands, refer to the section which describes that command. The SIF ASSOCIATIVE VALUES command must appear in the SIF file immediately after an IDENTIFIER command. (See Section 7.8.) These two commands, as a set, may never appear in a SIF file under the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OF)
- o before an INCLUDE TEXT command (INC/)

#### 10.4.5 ASSOCIATIVE VALUES Command Examples

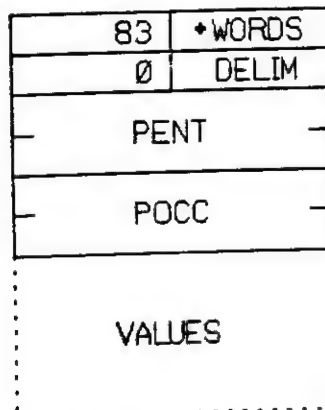
This section presents examples of the SIF ASSOCIATIVE VALUES command. Both valid and invalid forms are given and the ASCII form is always used.

In the following example, an entity 2 defines the SIF feature and the entity 3 is actually attached to the graphic element.



#### 10.4.2 ASSOCIATIVE VALUES Command - Binary Form

The binary form of the ASSOCIATIVE VALUES command is indicated by a command type of 83 in the command header. The number of words to follow field contains the number of Integer\*4 words needed to complete the command definition. The command value field contains the delimiter to be used when defining attribute data. The command subtype field is zero.



The delimiter character must be specified if attribute data is specified. The parent entity number and parent occurrence number are required even if set to 0. If the values field is omitted, an entity with all default values is inserted. The values field must be blank padded to an even 32-bit word.

#### 10.4.3 ASSOCIATIVE VALUES Command - Interface Form

The interface form allows you to generate an ASSOCIATIVE VALUES command by specifying a delimiter, parent entity number, parent occurrence number, number of characters in the attribute data, and the attribute data. The interface subroutine formats the command and writes it to a SIF output file. The following is an example of the calling sequence used when generating a SIF ASSOCIATIVE VALUES command via the interface library:

```
CALL CMD83 (delim, pent, pocc, nc, values)
```

delim - Byte character indicating delimiter for the values array.

pent - Integer\*2 parent entity number.

pocc - Integer\*2 parent occurrence number.

The range of the drawing file can be from a low of -2147483648 UORs for the x,y, and z coordinates to a high of +2147483647 UORs for the x,y, and z coordinates. This field is also optional. If you have specified no negative numbers allowed (NE=N in the environment file), then the low-high range may be from 0 to 4294967295 UORs for the x,y, and z coordinates. If the file is a two-dimensional file, only x and y coordinates need be specified.

The drawing units field indicates values for master units, positional units, and subunits and is optional. The description of the positional units and subunits is also recognized and can be any two ASCII characters.

The identification field can be from 1 to 1000 characters and is optional. Any ASCII character is valid.

#### 10.5.1 DRAWING IDENTIFICATION Command - ASCII Form

The ASCII form of the DRAWING IDENTIFICATION command is indicated by a command type of DID followed by a slash "/". The slash is then followed by the necessary and optional keywords. An example of the SIF DRAWING IDENTIFICATION command is as follows:

DID/NA=name, DA=date, MO=mode, RA=range, DU=drawing, units,  
ident

The NA keyword indicates the drawing file name and is required.

The DA keyword indicates the file creation date and is optional.

The MO keyword indicates the file dimension and is optional.

The RA keyword indicates the drawing file low-high range and is optional.

The DU keyword indicates the drawing units and is optional.

The ident field indicates the free-field drawing identification and is optional.

```

IDE/AS=1, CO=0, ID=2, KE=0
ASV/DE=!, ID=1, KE=0, !1!14!2!Complex shape!
TLC/CO=17, JU=3
TXT/OR=500, 9000, TH=500, TW=500, 14. Complex shape
IDE/AS=1, CO=0, ID=3, KE=0
ASV/DE=!, ID=2, KE=0, !1!Begin complex string!2!BST!3!50!
4!Complex shape!
BST/SO
ARC/P1=4000, 4000, P2=5000, 3000, P3=6000, 4000
EAR/CE=6000, 4500, P1=8000, 4500, P2=6000, 5000, ST=270, SW=180
ARC/P1=6000, 5000, P2=5000, 6000, P3=4000, 5000
EAR/CE=4000, 4500, P1=6000, 4500, P2=4000, 5000, ST=90, SW=180
EST/

```

The following example is invalid because the ASSOCIATIVE VALUES command was not used in conjunction with an IDENTIFIER command:

```

ASV/DE=!, ID=1, KE=0, !1!12!2!Conic!
TLC/CO=9, JU=3
TXT/OR=500, 9000, TH=500, TW=500, 12. Conic

```

The following example inserts an entity 1 into the database and automatically associates the SIF symbol with the linkage. The symbol is placed on level 60 using the next available graphic group number:

```

OVR/60
ASC/-1
LAC/LT=1
IDE/AS=1, CO=0, ID=1, KE=0
ASV/DE=!, ID=0, KE=0, !1!60!
SYM/OR=0, 0, FRAME

```

#### 10.5 DRAWING IDENTIFICATION Command

The SIF DRAWING IDENTIFICATION command can be used to define the drawing, file name, the date the file was created, the graphic mode (2-D/3-D), the range of the drawing file, the drawing units, and a free field identification. This command is recognized only as the first command in a SIF file.

The drawing file name can be from 1 to 12 characters and is required. The ASCII characters A-Z and 0-9 are valid. If the file name is more than nine characters, only the first nine characters are used to form the SIF disk file name.

The date can be from 1 to 8 characters and is optional. Any ASCII character is valid.

The graphic mode can be either 2 or 3 and is optional.

nc1     - Integer\*2 number of characters in file name  
 name    - Byte array containing file name  
 date    - 8-byte array containing date (free-form)  
 nc2     - Integer\*2 number of characters in identification  
 ident   - Byte array containing identification  
 gmode   - Integer\*2 graphic mode (2 for 2-D, 3 for 3-D)  
 mudf    - Integer\*4 number of master units in the design  
          file  
 sumu    - Integer\*4 number of subunits per master unit  
 pusu    - Integer\*4 number of positional units per subunit  
 mudesc   - 2-byte array containing description of master  
          units  
 sudesc   - 2-byte array containing description of subunits  
 grange   - Integer\*4 array containing the vertices defining  
          the lower and upper bounds of the graphic file.  
          The coordinates are entered in the order xlow,  
          ylo, xhigh, and yhigh.

An alternate form of the DRAWING IDENTIFICATION command is available. The form is called using a drawing name, date and identification. The following is an example of the alternate form of the DRAWING IDENTIFICATION command:

```
CALL CMD84(nc1,name,date,nc2,ident)
```

nc1     - Integer\*2 number of characters in the file name  
 name    - Byte array containing ASCII file name  
 date    - Byte array containing ASCII date. If a date is  
          entered, eight characters are required. A binary  
          0 indicates no date.  
 nc2     - Integer\*2 number of characters in identification.  
 ident   - Byte array containing identification. If "nc2"  
          is 0, "ident" may be a binary 0.

### 10.5.2 DRAWING IDENTIFICATION Command - Binary Form

The binary form of the DRAWING IDENTIFICATION command is indicated by a command type of 84 in the command header. The number of words to follow field contains the number of Integer\*4 words needed to complete the command definition. The command subtype field indicates the long form of the DRAWING IDENTIFICATION command and the command value field indicates the dimension of the drawing.

84	*WORDS
8	GMODE
NAME	
DATE	
IDENT	

84	*WORDS
1	GMODE
NAME	
DATE	
MUDF	
SUMU	
PUSU	
MUDESC SUDESC	
XLDW	
YLDW	
XHIGH	
YHIGH	
IDENT	

### 10.5.3 DRAWING IDENTIFICATION Command - Interface Form

The interface form allows you to generate a DRAWING IDENTIFICATION command that is then formatted and written to a SIF output file. The following is an example of the calling sequence used when generating a SIF DRAWING IDENTIFICATION command via the interface library:

```
CALL CMD84A(nc1,name,date,nc2,ident,gmode,mudf,
            sumu,pusu,mudesc,sudesc,grange)
```

ylo indicates the lowest y coordinate for the next graphic element

xhi indicates the highest x coordinate for the next graphic element

yhi indicates the highest y coordinate for the next graphic element

#### 10.6.2 RANGE Command - Binary Form

The binary form of the RANGE command is indicated by a command type of 85 in the command header. The number of words to follow field contains the number of Integer\*4 words needed to complete the definition. There is always four words to follow for 2-D files and six words to follow for 3-D files. The command subtype and value fields are always 0.

85	4
0	0
XLOW	
YLOW	
XHIGH	
YHIGH	

#### 10.6.3 RANGE Command - Interface Form

The interface form allows you to generate a RANGE command that is then formatted and written to an SIF output file. The following is an example of the calling sequence used to generate a SIF RANGE command:

CALL CMD85(low,high)

low - Integer\*4 array containing the coordinates of the lowest point of the graphic element

high - Integer\*4 array containing the coordinates of the highest point of the graphic element.

#### 10.5.4 DRAWING IDENTIFICATION Command Restrictions

The SIF DRAWING IDENTIFICATION command must appear in the SIF file as the first command. It will not be recognized if placed at any other location in the SIF file.

#### 10.5.5 DRAWING IDENTIFICATION Command Examples

This section presents examples of the SIF DRAWING IDENTIFICATION command. Both valid and invalid forms are given and the ASCII form is always used.

The following is an example of all keywords available in the DRAWING IDENTIFICATION command:

```
DID/NA=SIFDGN, DA=12/08/84, MO=2, RA=0,0, 350, 700,  
DU=4294967295, 1, 1, FT, 1N, SAMPLE ELLIPSE FILE
```

The following is an example of the alternate DRAWING IDENTIFICATION command:

```
DID/NA=CIRC, DA=12/07/58, SAMPLE CIRCLES
```

The following is an example of an illegal DRAWING IDENTIFICATION command because the NA keyword is not specified:

```
DID/DA=10/12/80, MO=3, TEST 3-D FILE
```

#### 10.6 RANGE Command

The SIF RANGE command is used to illustrate the low and high range of the graphic element which immediately follows. For 2-D files, an x and y low coordinate and x and y high coordinate is specified. For 3-D files, a z low and high coordinate is specified also. All coordinates are given in UORs. The SIF RANGE command has no effect in SIF-in processors and is generated during SIF-out processing only if the RA keyword is equal to "Y" in the environment file.

##### 10.6.1 RANGE Command - ASCII Form

The ASCII form of the RANGE command is indicated by a command type for RNG followed by a slash "/". The slash is then followed by the low/high coordinates. The x,y low coordinates are specified first while the x,y high coordinates immediately follow. If the SIF file is a 3-D file, the x,y,z low coordinates are specified first followed by the x,y,z high coordinates. An example of the SIF RANGE command is as follows:

```
RNG/xlo,ylo,xhi,yhi
```

xlo indicates the lowest x coordinate for the next graphic element

## 10.7 CHANGE Command

The SIF CHANGE command is used to indicate when graphic and database information is new and/or modified. The SIF CHANGE command is recognized during SIF processing if TC=Y is in the environment file; otherwise, the CHANGE command is ignored. The CHANGE command always appears immediately before the database or graphic definition. If changes are tracked, the new and modified bits are cleared in database and graphic elements created during the SIF-in process. If changes are not tracked, all elements created during the SIF-in process are considered to be new elements. The CHANGE command applies to 2-D files only.

### 10.7.1 CHANGE Command - ASCII Form

The ASCII form of the CHANGE command is indicated by a command type of CHA followed by a slash "/". The slash is then followed by a combination of keywords which specify the changes made to the database or graphic element. Examples of the SIF CHANGE command are as follows:

```
CHA/DB,NM  
CHA/DB,NE  
CHA/DB,MO  
CHA/GR,NE  
CHA/GR,MO  
CHA/GR,NM
```

The DB keyword indicates that the change was made to a DMRS database element. The GR keyword indicates that the change was made to an IGDS graphic element. The NE keyword indicates that the element is new. The MO keyword indicates that the element is modified. The NM keyword indicates that the element is new and modified.

### 10.7.2 CHANGE Command - Binary Form

The binary form of the CHANGE command is indicated by a command type of 86 in the command header. The number of words to follow is always 0 while the command subtype and command value indicates the element types and changes to be made.

86	0
STYPE	VALUE



#### 10.6.4 RANGE Command Restrictions

The restrictions of the RANGE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The RANGE command may appear anywhere in the SIF file except for the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OFF)
- o before an INCLUDE TEXT command (INC/)
- o before a CONTINUE command (CON/)

#### 10.6.5 RANGE Command Examples

This section presents examples of the SIF RANGE command where the ASCII form is always used.

The following is an example of a RANGE command describing the range of a 2-D circle whose origin is at the point 0,0 and radius is 100 UORs.

```
RNG/-100,-100,100,100  
CIR/CE=0,0,RA=100
```

The following is an example of the RANGE command describing the range of a 3-D text element.

```
RNG/100,100,100,1000,200,100  
TLC/CD=9,JU=3  
TXT/OR=100,100,100 TH=100,TW=100,AN=0.,TEST CASE
```

- o before an INCLUDE TEXT command (INC/)
- o before a CONTINUE command (CON/)

#### 10.7.5 CHANGE Command Examples

This section presents examples of the SIF CHANGE command. The ASCII form is used in each case.

The following example sets the modified bit in the DMRS database element that is created:

```
CHA/DB,MO
IDE/AS=1,ID=3
ASV/DE=?,ID=2,?1?NEW?2?#4#?3?
```

The next example sets the new and modified bits in an elliptical arc element:

```
CHA/GR,NM
EAR/CE=0,500,P1=100,500,P2=0,600,ST=30.,SW=95.
```

The next example is invalid because the CHANGE command should be immediately before the element it is describing.

```
CHA/GR,MO
MID/ON
IDE/AS=4,ID=19,AN ASCII USER DATA
IDE/AS=4,ID=19,WILL BE ATTACHED
IDE/AS=4,ID=19,TO THE GRAPHICS ELEMENT
IDE/AS=4,ID=19,THAT FOLLOWS
MID/OF
TXT/OR=200,-200,AN=90.,TEST CASE
```

stype - 1 indicates a change to a database element  
           2 indicates a change to a graphics element  
  
 value - 1 indicates that an element is new  
           2 indicates that an element is modified  
           3 indicates that an element is new and modified

### 10.7.3 CHANGE Command - Interface Form

The interface form of the CHANGE command allows you to generate a CHANGE command that is then formatted and written to a SIF output file. The following is an example of the calling sequence used to generate a SIF CHANGE command:

```
CALL CMD86(stype,value)
```

stype - Integer\*2 flag indicating whether the change was made to a graphic or database element. A value of 1 indicates a change to a database element. A value of 2 indicates a change to a graphics element.  
  
 value - Integer\*2 flag indicating the type of change which was made. A value of 1 means that an element was added. A value of 2 indicates that the element was modified. A value of 3 means that the element was new and modified.

### 10.7.4 CHANGE Command Restrictions

The restrictions of the CHANGE command are described in this section. For a complete description of conflicting commands, refer to the section which describes that command. The SIF change should always appear immediately before the database or graphics element to which it applies. The SIF CHANGE command may not appear in the following conditions:

- o between a MULTIPLE IDENTIFIER ON (MID/ON) and a MULTIPLE IDENTIFIER OFF command (MID/OFF)
- o between a GENERATE PARAGRAPH (PAR/) and a CLOSE PARAGRAPH command (CLP/)
- o between a BEGIN COMPLEX STRING (BST/) and an END COMPLEX STRING command (EST/)
- o immediately before an ASSOCIATIVE VALUES command (ASV/)
- o between a PATTERNING ON (PTN/) and a PATTERNING OFF command (PTN/OFF)

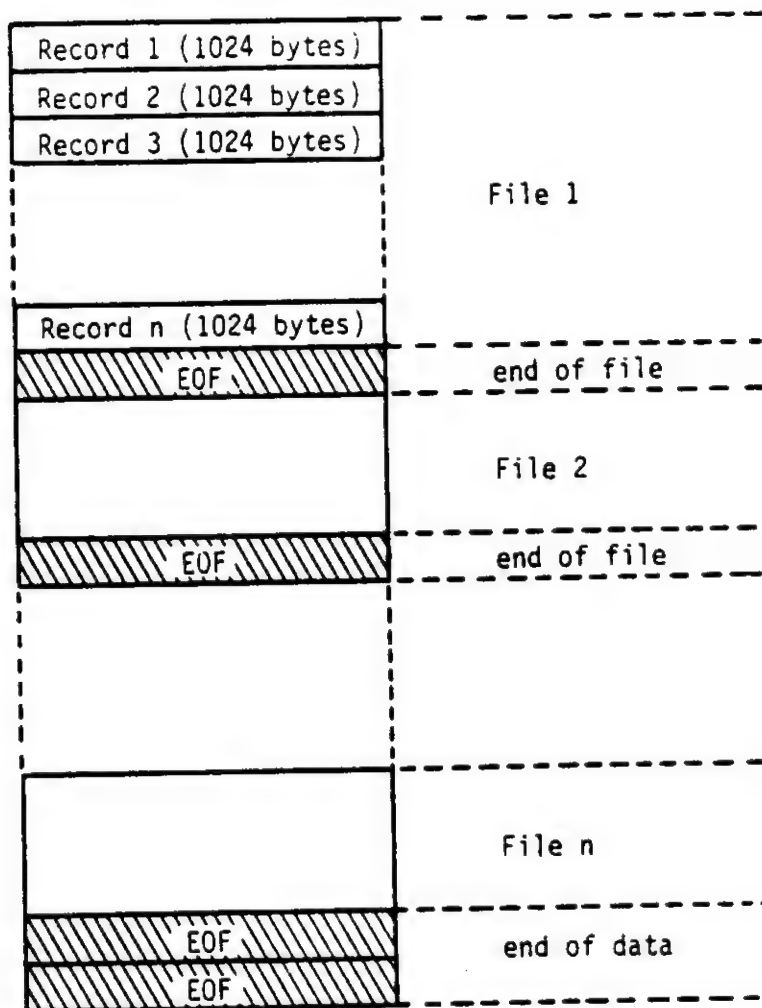


Figure 11-1. SIF Tape Format for ASCII and Binary Files

## 11. SIF TAPE FORMATS

SIF ASCII or binary command files can be transferred between systems via a SIF tape. The SIF tape format for ASCII and binary files is presented in Figure 11-1. SIF tapes have the following characteristics:

- o Unlabeled
- o 9-track
- o Odd parity
- o 800 to 1600 BPI
- o 1024 bytes per physical record
- o Single EOF between SIF files
- o Double EOF indicates end of tape
- o Multiple volumes not allowed

The SIF ASCII tape file consists of 1024 byte physical records. Each physical record consists of fixed length logical records. The logical record length can be from 20 to 80 bytes and must be consistent throughout the tape. You must decide which logical record length is the most efficient. The logical record length is defined by the LR keyword in the SIF environment file. (See Section 2.16.) The number of logical records per physical record is determined by the following equation:

$$\text{NUMLOG} = 1024/\text{LOGLEN}$$

In the above equation, NUMLOG is the number of logical records per physical record, and LOGLEN is the specified logical record length. Any remaining portion of the physical record should be padded with blanks. SIF ASCII commands must begin on a logical record boundary, and any unused portion of the logical record must be padded with blanks.

The SIF binary tape file consists of 1024 byte physical records. Each physical record consists of variable length logical records. Each logical record is one SIF binary command. SIF binary commands have a range of 4 to 1024 bytes and are on 32-bit word boundaries. Since SIF binary commands contain the command size, logical binary records can cross physical record boundaries. The last physical record on each binary tape file must be padded to 1024 bytes with the SIF PAD command. (See Section 10.1.)

IN-2

## INDEX

ATI processor 2-3, 2-16, 2-21, 2-23, 2-28, 2-29, 2-31, 2-32  
 ATO processor 1-3, 2-16, 2-21, 2-23, 2-32  
 attribute data 10-7 through 10-10  
 attribute linkage 2-7  
 attribute values 2-11  
 BBC/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-91 through 8-93, 8-96,  
     8-100, 8-111, 8-112  
 BBS/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-100 through 8-102,  
     8-105, 8-110 through 8-112  
 BEGIN B-SPLINE CURVE command 8-89 through 8-93, 8-100  
 BEGIN B-SPLINE SURFACE command 8-100 through 8-105, 8-110  
 BEGIN COMPLEX STRING command 8-24, 8-36, 8-49, 8-51, 8-52,  
     8-53, 8-56, 8-61, 8-67, 8-73, 8-80, 8-86, 8-92, 8-105, 9-5,  
     9-15, 10-2, 10-4, 10-10, 10-17, 10-19  
 BEGIN PARAGRAPH command 8-18, 8-24  
 BEGIN PATTERNING command 8-24  
 BEGIN SOLID command 8-66, 8-67, 8-69  
 BEGIN STRING command 2-29  
 BEGIN SURFACE command 8-60, 8-61  
 BEGIN SYMBOL command 2-6, 2-9, 2-11, 8-54 through 8-57, 8-59,  
     8-61, 8-67, 8-73, 8-80, 8-86, 8-92, 8-105  
 binary file 1-4, 1-5, 2-30, 3-1, 6-1, 6-2  
 BINLIB 6-1  
 BOU 8-101, 8-111  
 boundary 11-1  
 BOUNDARY command 8-104, 8-111, 8-112  
 boundary elements 8-110, 8-111  
 boundary number 8-111, 8-112



/AB switch 2-27

active pattern 7-17

active Z 7-1, 7-7

ACTIVE Z command 7-4, 7-5, 7-6

angle of rotation 9-14

arc 7-3, 7-7, 7-50, 8-14, 8-52

ARC command 2-1, 2-3, 2-22, 8-13 through 8-18

area pattern number 7-25, 7-26

area patterns 7-17, 7-22, 7-24

APT processor 2-10, 2-30

AR keyword 2-1

AS keyword 2-3

ASCII command file 4-1

ASCII file 1-3, 2-3, 3-1, 6-1 through 6-3

ASCLIB 6-1

ASI processor 1-1, 2-3, 2-10, 2-14, 2-30

ASO processor 1-1, 2-3, 2-21, 2-23, 2-30

association 7-1

ASSOCIATION command 7-11 through 7-13

association type 7-43 through 7-47, 7-51

ASSOCIATIVE VALUES command 7-3, 7-6, 7-9, 7-13, 7-15, 7-20,  
7-26, 7-32, 7-37, 7-41, 7-48, 8-4, 8-11, 8-18, 8-24, 8-31,  
8-36, 8-43, 8-48, 8-52, 8-56, 8-61, 8-67, 8-73, 8-80, 8-87,  
8-92, 8-105, 9-5, 9-15, 10-2, 10-4 through 10-11, 10-17,  
10-19

ASV/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-105

ATG processor 1-1, 1-3, 2-3, 2-5, 2-9, 2-10, 2-11, 2-12, 2-14,  
2-17, 2-18, 2-20, 2-25, 2-28, 2-29, 2-31, 7-17

circle element 7-3, 7-22  
 circle line string 7-7  
 circular arc 8-1  
 circular truncated cone 8-1, 8-87, 8-89  
 CIRCULAR TRUNCATED CONE command 8-82 through 8-87  
 circular truncated cone element 8-82  
 classification 7-1  
 CLASSIFICATION command 7-7 through 7-9, 8-49, 8-52, 8-54, 8-56  
 CL keyword 2-5  
 clockwise sweep 8-15  
 closed 8-90, 8-91, 8-92, 8-101, 8-102  
 closed ellipse element 8-7  
 closed line string 7-7, 8-2  
 closed representing a hole 8-91, 8-92, 8-101  
 CLOSE PARAGRAPH command 7-3, 7-6, 7-9, 7-12, 7-15, 7-20, 7-26,  
 7-32, 7-37, 7-41, 7-46, 7-49, 8-4, 8-11, 8-18, 8-24, 8-26,  
 8-29, 8-31, 8-36, 8-43, 8-48, 8-52, 8-56, 8-61, 8-67, 8-73,  
 8-80, 8-87, 8-92, 8-105, 9-5, 9-10, 9-15, 9-16, 9-19, 9-21,  
 10-2, 10-4, 10-10, 10-17, 10-19  
 CLP/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-105  
 code number 10-8  
 CD keyword 2-9  
 color 7-1  
 complex element 8-44  
 complex shape 8-51, 8-53  
 complex solid 8-1  
 complex string 7-9, 8-1, 8-14, 8-49, 8-51  
 COMPLEX STRING command 8-20, 8-56

BSF 8-60, 8-61

BSO/ 8-66, 8-67

B-spline curve 8-1, 8-90, 8-100, 8-113

B-spline curve set 8-111

B-SPLINE CURVE command set 8-61, 8-67, 8-73, 8-80, 8-87, 8-89,  
8-92, 8-112, 8-113

B-spline surface 8-1, 8-107 through 8-109, 8-111

B-SPLINE SURFACE command set 8-61, 8-67, 8-73, 8-80, 8-87,  
8-92, 8-104, 8-105, 8-112, 8-113

B-spline surface element 8-101

BST/ 8-61, 8-67, 8-73, 8-80, 8-86, 8-92, 8-105

BSY/ 8-73, 8-80, 8-86, 8-92, 8-105

BTI processor 1-3, 2-4, 2-18, 2-23, 2-30, 2-32

BTD processor 1-3, 2-4, 2-18, 2-23, 2-30, 2-32

BY keyword 2-4

byte storage 2-4

capped surface element 8-66

cell 8-26, 8-57

cell description 8-56

cell library 2-8, 2-9, 2-11, 2-25, 2-27, 2-30, 8-21, 8-22,  
8-54, 8-55, 8-57, 8-59

cell name 8-21, 8-22

cell origin 8-21, 8-22

CHANGE command 2-31, 7-3, 7-6, 7-9, 7-13, 7-15, 7-20, 7-26,  
7-41, 7-50, 10-18, 10-19, 10-20

characteristics commands 7-1

circle 7-50, 8-1, 8-57, 10-17

CIRCLE command 2-22, 8-7 through 8-13

database 1-4, 7-1, 10-7, 10-11, 10-18, 10-19  
database linkages 7-46, 7-47, 7-48, 7-50  
DB keyword 2-9  
DE keyword 2-10  
DEC format 2-4, 2-5, 2-18, 2-19  
delimiter 8-28, 10-7 through 10-10  
delimiter character 8-26, 8-27  
design file 1-3, 2-11 through 2-14, 2-16, 2-17, 2-21, 2-23,  
2-25, 2-26, 2-30, 2-31, 3-1, 8-55, 8-59  
design plane 5-2  
DF keyword 2-10, 2-23  
disjointed 8-72  
disjointed point string 8-70, 8-71  
displayable attribute 9-11  
displayable attribute number 9-14, 9-16  
DLT processor 1-3, 2-3, 2-10, 2-30  
DM keyword 2-11  
DMRS database 1-4, 1-5, 2-9, 2-10, 2-31, 7-48, 10-20  
DP keyword 2-11  
DRAWING IDENTIFICATION command 10-11 through 10-15  
drawing units 10-12  
DU keyword 2-12  
EBC/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-93, 8-96, 8-100,  
8-111  
EBCDIC 2-16  
EBS/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-101, 8-105, 8-110,  
8-111  
ED keyword 2-13

complex surface 8-1

CON/ 8-67, 8-80, 8-87, 8-92, 8-105

cone as a solid 8-88

cone as a surface 8-90

conic 8-1, 10-5

CONIC command 8-47 through 8-49, 10-4, 10-6

construction class elements 7-9

contiguous file 2-5, 2-9, 2-11, 2-17, 2-25, 2-29

continuation line 8-1

CONTINUE command 6-4, 6-5, 7-3, 7-6, 7-9, 7-12, 7-15, 7-20,  
7-26, 7-32, 7-37, 7-41, 7-46, 7-48, 7-49, 8-3, 8-11, 8-18,  
8-24, 8-30, 8-31, 8-36, 8-43, 8-48, 8-49, 8-52, 8-56, 8-61,  
8-67, 8-80, 8-87, 8-92, 8-105, 9-5, 9-15, 10-2, 10-4 through  
10-7, 10-17, 10-20

continuous 8-72

continuous point string 8-70, 8-71

CPC processor 1-3, 1-5, 2-5, 2-8, 2-9, 8-54, 8-55

/CR switch 2-9

create symbol 8-56

CTC 8-82, 8-87, 8-89

curve 6-5, 8-1, 8-30, 10-5

CURVE command 8-30 through 8-32, 10-4, 10-6

CYL 8-76, 8-80

cylinder 8-1

cylinder as a solid 8-81

cylinder as a solid surface 8-83

CYLINDER command 8-76 through 8-80

CZ keyword 2-9

EST/ 8-60, 8-61, 8-65, 8-67, 8-73, 8-80, 8-86, 8-92, 8-105  
 ESY/ 8-61, 8-73, 8-80, 8-86, 8-92, 8-105  
 ET keyword 2-16  
 exceptions file 2-8  
 /EX switch 2-9  
 FE keyword 2-16  
 FENCE processor 1-4, 2-14, 2-16  
 FL keyword 2-17  
 font 7-1  
 FONT command 7-14, 7-15, 8-54, 8-56  
 /FS switch 2-9, 2-20  
 FS keyword 2-9, 2-17  
 FT keyword 2-18  
 /FU switch 2-15, 2-28  
 GENERATE ARC command 8-49, 8-52  
 GENERATE CONIC command 8-49, 8-52  
 GENERATE CURVE command 8-49, 8-52  
 GENERATE ELLIPTICAL ARC command 8-49, 8-52  
 GENERATE LINE STRING command 8-49, 8-52  
 GENERATE PARAGRAPH command 4-3, 7-3, 7-6, 7-9, 7-12, 7-15,  
 7-20, 7-26, 7-31, 7-37, 7-41, 7-46, 7-49, 8-4, 8-11, 8-31,  
 8-36, 8-43, 8-48, 8-52, 8-56, 8-61, 8-67, 8-73, 8-80, 8-87,  
 8-92, 8-105, 9-5, 9-10 through 9-16, 9-19, 10-2, 10-4, 10-10,  
 10-17, 10-19  
 GENERATE PARAGRAPH LINE command 9-10, 9-16 through 9-20  
 GENERATE SYMBOL command 4-2, 8-26, 8-29  
 GENERATE TEXT command 8-28  
 GENERATE TEXT LINE command 4-3, 8-26

edit delimiter 9-16, 9-18  
 /EF switch 2-8  
 element class 7-1  
 EL keyword 2-14, 7-2, 7-4  
 ellipse 7-11, 7-41, 8-1  
 ELLIPSE command 8-34 through 8-37  
 ellipse elements 7-3  
 elliptical arc 7-3, 7-41, 8-1, 8-44  
 ELLIPTICAL ARC command 2-22, 8-40 through 8-44  
 elliptical arc element 10-20  
 elliptical arc origin 8-41  
 EM keyword 2-14  
 END B-SPLINE CURVE command 8-89, 8-92, 8-100  
 END B-SPLINE SURFACE command 8-101, 8-104, 8-110  
 END COMPLEX STRING command 8-11, 8-24, 8-36, 8-52, 8-54, 8-61,  
     8-67, 8-73, 8-80, 8-86, 8-92, 8-105, 9-5, 9-15, 10-2, 10-4,  
     10-10, 10-17, 10-19  
 END SOLID command 8-67, 8-69  
 END SURFACE command 8-61, 8-65  
 END SYMBOL command 2-6, 2-14, 8-29, 8-56, 8-59, 8-61, 8-67,  
     8-73, 8-80, 8-86, 8-92, 8-105  
 enter data field 2-13, 2-14, 2-21, 8-27, 8-28, 8-29, 9-18  
 enter data fill-in 8-26  
 entity 10-7  
 entity number 7-45, 10-8, 10-9  
 environment file 1-1, 1-3, 1-4, 2-3, 2-4, 2-9, 2-12 through  
     2-16, 2-18, 2-22 through 2-24, 2-26, 2-29, 2-31, 2-32, 2-34,  
     3-1, 7-2, 7-4, 7-30, 8-57, 8-59, 10-12, 10-15  
 ESO 8-66, 8-67, 8-69

IGDS element class 7-7, 7-8

IGDS ellipse 8-34

IGDS ELLIPSE 8-1

IGDS ellipse element 8-7

IGDS font 7-14, 7-16

IGDS font library 2-17

IGDS justification 7-30

IGDS level 7-1, 7-2, 7-4

IGDS LINE, LINE STRING, SHAPE 8-1

IGDS PARTIAL ELLIPSE 8-1

IGDS partial ellipse element 8-40

IGDS POINT STRING 8-1

IGDS point string element 8-70

IGDS SURFACE 8-1

IGDS text node 7-34

IGDS text node lines 7-34

INC/ 8-67, 8-73, 8-87, 8-92, 8-105

INCLUDE TEXT command 2-13, 2-14, 7-3, 7-6, 7-9, 7-12, 7-15,  
7-20, 7-26, 7-32, 7-37, 7-41, 7-46, 7-49, 8-4, 8-11, 8-18,  
8-26 through 8-31, 8-36, 8-43, 8-48, 8-52, 8-56, 8-61, 8-67,  
8-73, 8-80, 8-87, 8-92, 8-105, 9-15, 10-10, 10-17, 10-20

index file 1-1, 1-3, 2-15, 2-30, 3-1

IN keyword 2-18

integer storage 2-18

interface libraries 6-1 through 6-4, 6-6

IX keyword 2-20

justification 4-3, 7-1, 7-30, 7-33, 7-34, 7-37, 9-6



GRAPHIC ASSOCIATION DESCRIPTOR command 10-2 through 10-4  
 GRAPHIC ELEMENT GENERATION command 8-54, 8-56  
 graphic grouping 7-12  
 graphic group number 2-25, 2-26, 7-11 through 7-13, 10-2  
 through 10-4, 10-11  
 group data element 10-2  
 GTA processor 1-4, 2-1, 2-3, 2-5, 2-9, 2-10, 2-13, 2-16, 2-17,  
 2-18, 2-22, 2-24, 2-25, 2-29, 2-31, 2-33, 2-34  
 HLP processor 1-4, 2-14, 2-30  
 hole 8-2, 8-4, 8-8, 8-10, 8-11, 8-35, 8-36, 8-51, 8-90, 8-93,  
 8-102  
 identifier 7-1  
 IDENTIFIER command 2-31, 7-43, 7-45, 7-46, 7-48, 10-7, 10-10,  
 10-11  
 IGDS ARC 8-1  
 IGDS B-SPLINE CURVE 8-1  
 IGDS B-SPLINE SURFACE 8-1  
 IGDS CAPPED SURFACE 8-1  
 IGDS cell 1-3, 2-6, 8-21  
 IGDS CELL 8-1  
 IGDS cell library 1-3, 2-5, 2-6  
 IGDS circular arc 8-13  
 IGDS circular arc element 8-14  
 IGDS CIRCULAR TRUNCATED CONE 8-1  
 IGDS COMPLEX STRING OR SHAPE 8-1  
 IGDS CONIC 8-1  
 IGDS CURVE 8-1  
 IGDS design file 1-4, 2-9, 2-10, 7-2, 7-7, 7-11, 7-14, 7-17,  
 7-23

map index file 2-20, 2-21

MASK 2-7, 8-55

masked symbol 8-56

MASK user command 2-7

message file 2-14, 2-15, 2-18, 7-39, 8-54

mirrored 8-22, 9-11

mirroring 4-2, 6-5, 8-21, 8-23, 9-1 through 9-4, 9-14

/MK switch 2-7

MO keyword 2-23

MT keyword 2-23

multiple identifier 7-1

MULTIPLE IDENTIFIER command 7-47 through 7-49, 7-50, 8-4

MULTIPLE IDENTIFIER OFF command 8-4, 8-11, 8-17, 8-31, 8-36,  
8-43, 8-48, 8-52, 8-56, 8-61, 8-67, 8-72, 8-80, 8-86, 8-92,  
9-1, 9-15, 10-2, 10-4, 10-10, 10-17, 10-19

MULTIPLE IDENTIFIER ON command 8-4, 8-11, 8-17, 8-31, 8-36,  
8-43, 8-48, 8-52, 8-56, 8-61, 8-67, 8-72, 8-80, 8-86, 8-92,  
9-1, 9-15, 10-2, 10-4, 10-10, 10-17, 10-19

multiple linkages 7-1, 7-48, 7-49

MULTIPLE LINKAGE OFF command 7-3, 7-6, 7-9, 7-12, 7-15, 7-20,  
7-26, 7-32, 7-37, 7-41, 7-49, 8-24

MULTIPLE LINKAGE ON command 7-3, 7-6, 7-9, 7-12, 7-15, 7-20,  
7-26, 7-32, 7-37, 7-41, 7-49, 8-24

multiple linkage user data 8-52

multi-segmented patterning 7-17

/NA switch 2-10

NDX processor 1-4, 2-1, 2-14, 2-23, 3-1

nonedit delimiter 9-16, 9-18, 9-19

normal symbol 8-56

KND/ 8-96, 8-100, 8-112  
 KNOT command 8-89, 8-92, 8-101, 8-104, 8-112, 8-113  
 knot elements 8-110  
 knots 8-91, 8-92, 8-102  
 levels 2-24, 7-3, 8-4, 8-43, 8-52  
 LINE/AREA CHARACTERISTICS command 8-49, 8-52, 8-54, 8-56  
 linear pattern number 7-25  
 linear patterns 7-17, 7-24, 7-26  
 line characteristics 7-1  
 LINE CHARACTERISTICS command 2-25 through 2-27, 7-23 through 7-26  
 line colors 7-23, 7-25, 7-26  
 line element 7-20, 7-47, 8-2  
 line spacing 7-1, 7-34 through 7-37  
 line string 6-5, 7-3, 7-11, 7-26, 7-50, 7-51, 8-1, 8-2, 8-4, 8-52, 8-57, 10-5  
 LINE STRING command 8-1 through 8-4, 8-7, 10-4, 10-6  
 line string elements 8-2, 8-4  
 line style 7-1, 7-23, 7-25, 7-26, 8-4, 8-52  
 line type 8-3  
 line weights 7-23, 7-25, 7-26  
 linkage 7-43  
 linkage family class 7-43, 7-45, 7-46  
 logical record 11-1  
 LR keyword 2-21  
 LST/ 10-6  
 MA keyword 2-22

pattern scale 7-18 through 7-20, 7-25, 7-26  
pattern seed file 7-24  
pattern set 2-28  
pattern type 7-18, 7-19, 7-20, 7-22  
PDV processor 2-25, 7-17, 7-21  
PF keyword 2-25 through 2-27, 7-24  
physical record 11-1  
point string 8-1  
POINT STRING command 8-70 through 8-72  
POL 8-93, 8-96, 8-101, 8-114, 8-115  
POLE command 8-89, 8-92, 8-104, 8-114, 8-115  
pole elements 8-110  
poles 8-112, 8-116  
primary axis 8-34, 8-36, 8-40, 8-41  
primary class elements 7-9  
processor 1-1  
protection counter 7-44  
PST 8-70  
PT keyword 2-28  
radius 8-7  
RA keyword 2-29  
RANGE command 2-29, 2-31, 7-3, 7-6, 7-9, 7-13, 7-15, 7-20,  
7-26, 7-41, 7-46, 7-50, 10-15 through 10-17  
RCV processor 1-4, 2-10, 2-30  
REC processor 2-3  
record boundaries 5-1

occurrence number 7-43 through 7-46, 10-7 through 10-9  
 open 8-90 through 8-92, 8-101, 8-102  
 open line string 8-4  
 order 8-91, 8-92, 8-112  
 origin 4-3  
 orphan cells 2-5, 2-7, 8-54, 8-55, 8-59  
 overlay 7-1  
 OVERLAY command 7-1 through 7-3, 8-49, 8-52, 8-54, 8-56  
 OV keyword 2-24  
 PAD command 5-1, 6-3, 10-1  
 PAR/ 8-61, 8-67, 8-73, 8-80, 8-87, 8-92, 8-105  
 paragraph characteristics 7-1  
 PARAGRAPH CHARACTERISTICS command 4-3, 7-35 through 7-37, 7-39,  
 8-54, 8-56, 9-11  
 PARAGRAPH command 2-12 through 2-14, 2-18, 2-22, 2-29, 2-33,  
 6-6, 7-34, 7-35, 8-27, 9-21  
 pattern angle 7-18 through 7-20  
 PATTERN command 2-25, 2-27, 7-17 through 7-20, 7-24, 8-13  
 pattern delta 7-18  
 pattern file 2-25, 2-27  
 patterning 7-1  
 PATTERNING command 8-7, 8-20, 9-5  
 PATTERNING OFF command 9-5, 9-15, 10-2, 10-4, 10-10, 10-17,  
 10-19  
 PATTERNING ON command 9-5, 9-15, 10-2, 10-4, 10-10, 10-17,  
 10-19  
 pattern name 7-18 through 7-20  
 pattern number 7-24  
 patterns 1-1, 1-5, 7-1

SIFNEQ subroutine 6-4  
 SIFPTS subroutine 6-5, 6-6, 8-4, 8-31, 8-48, 8-72  
 SIF symbol set 2-6, 2-7, 2-9  
 SIFTND subroutine 6-6  
 SIFTXT subroutine 6-7  
 signed values 5-2  
 single-segmented patterning 7-17  
 solid 8-2, 8-4, 8-8, 8-10, 8-11, 8-35, 8-36, 8-51, 8-68, 8-76,  
 8-79, 8-84, 8-86, 8-87  
 SOLID command set 8-66  
 start angle 8-40, 8-41  
 STD format 2-4, 2-5, 2-18, 2-19, 2-20  
 STOP PATTERNING command 8-24  
 surface 8-64, 8-76, 8-79, 8-84, 8-86, 8-89  
 SURFACE command set 8-60, 8-112  
 sweep angle 8-14, 8-40, 8-41, 8-43  
 SX keyword 2-30  
 symbol 2-17, 8-1  
 SYMBOL 2-5, 2-22  
 SYMBOL command 2-6, 2-7, 2-9, 2-13, 2-14, 2-22, 8-21 through  
 8-27  
 tape file 11-1  
 TC keyword 2-31  
 TD keyword 2-32  
 temporary origin 7-1  
 TEMPORARY ORIGIN command 7-39 through 7-41, 8-56  
 text element 2-14, 2-21, 7-3, 7-9, 7-15, 7-30, 7-31, 7-32,  
 10-17

recovery file 1-4  
rotation 9-2  
rotation angle 4-2, 4-3, 9-1, 9-3, 9-4, 9-11, 9-14, 9-15  
rule lines 8-102  
scaled linear patterns 7-17, 7-20  
scale factor 4-2, 6-5, 7-17, 8-21  
SD keyword 2-23, 2-29  
secondary axis 8-34, 8-36, 8-40, 8-41, 8-44  
seedfile 2-30  
seed file 2-12, 2-20, 2-23, 2-29  
shape 2-21, 8-2  
shape elements 8-2  
SIFATT subroutine 6-4, 10-10  
SIFCLO subroutine 6-3  
SIFCOL subroutine 6-3  
SIFENV 2-1  
SIF environment file 2-1  
SIFEOF subroutine 6-3  
SIFERR processor 1-4, 2-1, 2-14, 2-15, 2-23  
SIF index file 1-4, 1-5  
SIFMAT subroutine 6-5, 8-23  
SIF message file 2-10  
SIFMOD subroutine 6-3  
SIFOPE subroutine 6-2  
SIF orphan cells 2-8  
SIFNDX 1-4

TRI processor 1-1, 1-3, 1-4, 2-5, 2-9 through 2-12, 2-14, 2-17,  
 2-18, 2-20, 2-25, 2-28 through 2-31, 7-17  
 TRD processor 1-5, 2-1, 2-5, 2-9 through 2-13, 2-16 through  
 2-18, 2-22 through 2-25, 2-29 through 2-31, 2-33, 2-34  
 truncate 1-5  
 truncated cone element 8-76  
 TRU processor 1-5, 2-3, 2-30  
 2-D B-Spline curve 8-94 through 8-98  
 TX keyword 2-34  
 TY keyword 2-34  
 U direction 8-102, 8-103, 8-113  
 unsigned 7-5, 7-6  
 unsigned values 5-2  
 UORs 2-13, 4-3, 5-2, 7-36, 7-37, 7-39, 8-7, 8-8, 8-10, 8-17,  
 8-22, 8-23, 8-36, 8-41, 8-47, 8-114, 9-1, 9-2, 9-11, 9-12,  
 10-12  
 user command 1-4, 2-16  
 user data 7-45, 7-46  
 USER DATA 10-20  
 user ID number 7-45  
 user linkage 2-7, 7-1, 7-43, 7-44, 7-46 through 7-48, 7-50,  
 7-51  
 V direction 8-102, 8-103, 8-113  
 WEI 8-93, 8-96, 8-101, 8-115  
 weight 7-1, 8-4, 8-116  
 WEIGHT command 8-89, 8-92, 8-101, 8-104, 8-115, 8-116  
 weight elements 8-110  
 weight factor 8-115, 8-116  
 working units 2-12, 2-13, 5-2



text font 7-1, 7-15, 9-8  
 text characteristics 7-1  
 TEXT CHARACTERISTICS command 2-28, 7-30, 7-31, 7-32  
 TEXT command 2-12 through 2-14, 2-18, 2-22, 2-28, 2-34, 7-14,  
 7-15, 7-30, 7-31, 8-27  
 text elements 7-13, 7-31  
 text height 4-3, 9-1, 9-2, 9-4, 9-11, 9-12, 9-14, 9-16  
 text justification 7-1, 7-30, 7-31, 7-32, 7-36  
 text length 7-1, 7-30  
 TEXT LINE CHARACTERISTICS command 8-54, 8-56  
 TEXT LINE command 6-7, 9-1 through 9-5  
 text lines 8-27  
 text node justification 7-35  
 text node lines 7-35 through 7-37, 7-39, 9-16, 9-18  
 text node maximum line length 7-1  
 text nodes 2-11 through 2-14, 2-17, 6-6, 7-37, 7-39, 8-26,  
 8-29, 9-11, 9-15, 9-16, 9-21  
 text node number 9-11, 9-14, 9-16  
 text origin 7-32, 9-4  
 TEXT STRING command 9-15  
 text strings 7-13, 8-26, 8-29, 8-57, 9-1  
 text width 4-3, 9-1, 9-2, 9-4, 9-11, 9-12, 9-14, 9-16  
 TF keyword 2-32  
 3-D B-Spline Curve 8-99  
 TN keyword 2-33  
 /TR 2-17  
 tracked 10-18

# Document User Questionnaire

This questionnaire provides the opportunity to help us improve documents for you, the user. After you respond to the items listed below, please fold and mail to the address on the reverse side. Postage is prepaid.

Document number: \_\_\_\_\_

Document title: \_\_\_\_\_

1. Are the scope and sequence logically presented?

\_\_\_\_\_

2. Is the information understandable? If not, please identify section(s), page(s), and subject(s).

Section: \_\_\_\_\_

Page: \_\_\_\_\_

Subject: \_\_\_\_\_

3. Are the figures presented correctly and in sufficient quantity?

\_\_\_\_\_

4. Are the examples helpful and of sufficient quantity? For what subject would more examples be helpful?

\_\_\_\_\_

5. Are specific subjects easily found? If not, please list the subject that was difficult to locate.

\_\_\_\_\_

Suggestions: \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Your position in your company \_\_\_\_\_

Length of experience with Intergraph system? \_\_\_\_\_ YRS. \_\_\_\_\_ MOS.

\_\_\_\_\_

INTERGRAPH



No Postage  
Necessary  
if Mailed in the  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 79 HUNTSVILLE, ALABAMA

Postage will be paid by addressee

**Intergraph Corporation**

Director of Technical Publications  
One Madison Industrial Park  
Huntsville, Alabama 35807-9985



# INTERGRAPH

---

Intergraph Corporation  
One Madison Industrial Park  
Huntsville, Alabama 35807  
Phone: (205) 772-2000  
TWX 810-726-2180

---

